NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MODELLING EXPERIMENTAL PROCEDURES
FOR MANIPULATOR CALIBRATION

by

William E. Swayze

December 1991

Thesis Advisor: Morris R. Driels

Approved for public release; distribution is unlimited

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release: Distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL *(If applicable)* ME	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS *(City, State and ZIP Code)* Monterey, CA 93943-5000	7b. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State, and ZIP Code)*	10. SOURCE OF FUNDING NUMBER PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
MODELING EXPERIMENTAL PROCEDURES FOR MANIPULATOR CALIBRATION

12. PERSONAL AUTHORS
WILLIAM E. SWAYZE

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Master's Thesis	FROM_____ TO_____	DECEMBER 1991	211

16. SUPPLEMENTARY NOTATION
The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block numbers)*
FIELD	GROUP	SUB-GROUP	robot manipulator calibration

19. ABSTRACT *(Continue on reverse if necessary and identify by block numbers)*

A six degree of freedom manipulator, a PUMA 560, is calibrated using three different measurement systems in order to improve the accuracy of the manipulator. Closed loop kinematic chain modeling theory is presented. Variations in the models for each calibration method are presented. A simulation study is conducted to determine feasibility of the proposed methods. Experimental data is obtained and the actual calibration performed. A comparative analysis between both simulation and experiment and between measurement systems is performed. Various aspects regarding measurement system modelling are discussed. The calibrated kinematic parameters are presented as results.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
XX UNCLASSIFIED/UNLIMITED __ SAME AS RPT __ DTIC USERS	unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL M.R. Driels	22b. TELEPHONE *(Include Area Code)* (408) 646-3383	22c. OFFICE SYMBOL ME/Dr

DD Form 1473, JUN 86 *Previous editions are obsolete.* S/N 0102-LF-014-6603

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Approved for public release; distribution unlimited.

Modelling Experimental Procedures for Manipulator Calibration

by

William Earl Swayze
Lieutenant, United States Navy
B.S., Auburn University, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1991

ABSTRACT

A six degree of freedom manipulator, a PUMA 560, is calibrated using three different measurement systems in order to improve the accuracy of the manipulator. Closed loop kinematic chain modeling theory is presented. Variations in the models for each calibration method are presented. A simulation study is conducted to determine feasibility of the proposed methods. Experimental data is obtained and the actual calibration performed. A comparative analysis between both simulation and experiment and between measurement systems is performed. Various aspects regarding measurement system modelling are discussed. The calibrated kinematic parameters are presented as results.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

There are two main objectives that are addressed in this thesis. The first objective addresses development of practical manipulator calibration methods. A number of different devices and techniques have been employed to calibrate manipulators. However, most methods involve highly sophisticated, delicate and expensive measurement systems which are well suited for laboratory work but are not practical in an industrial environment. The second objective addresses problems associated with modelling measurement systems within closed loop kinematic chains.

The goal of calibration is to improve the accuracy of the manipulator. Accuracy, in the sense used here, is the ability of a manipulator to achieve a commanded position and orientation, pose for short, of its end effector. The end effector pose is a function of both fixed geometric properties of the robot, such as link lengths, and variable geometric properties, such as angular displacement of a rotary joint. The kinematic model is developed from both the fixed and variable geometric properties and in a qualitative sense, these models are both well understood and well defined. However, errors between the pose predicted by a model and the pose achieved by a typical manipulator have been shown by

experiment to be 10 mm or more [Ref 1]. These errors are due, in most part, to differences between the design or nominal values of the geometric properties of the manipulator and the actual manufactured values.

Another measure of a manipulator's performance is its repeatability. Repeatability is the ability of a manipulator to achieve an identical pose each time it is commanded to a specific pose. Current experimentation shows that manipulators have a repeatability on the order of 0.3 mm [Ref 2]. Therefore, a measure of the success of calibration is a model with an accuracy which approaches the manipulator's repeatability.

There are four basic steps in the calibration process [Ref. 3] and these steps are described as follows:

- A closed chain kinematic model of the manipulator and measurement system is developed. During this process, identifiable parameters are determined and the measured quantity or quantities are specified. A set of error functions are derived from the difference in the measured quantities and the quantities predicted by the model. Nominal parameter values are provided by manipulator manufacturing specifications, measurement system specifications and the location of the measurement system.

- Next, experimental measurements are taken. These measurements are a function of the actual parameter values. Corresponding joint variable data is incorporated into the measurement set.

- Identification of the parameters is performed utilizing the experimental data. This process consists of systematically adjusting the nominal parameters until the model predictions match the experimental data and hence the error functions become zero.

- The final step involves incorporating the identified parameters into the software used to control the manipulator.

The first three steps were performed on a PUMA 560 six degree of freedom manipulator arm utilizing three different measurement systems. The first of these methods corresponds to a laboratory method and uses a Coordinate Measuring Machine for full pose measurement. This highly accurate method provides a benchmark for the other two methods. Although step four of the process is not performed, computer simulation of the process is conducted to quantify the success of the calibration method.

Although standardized and reliable approaches to kinematic modelling of manipulators exist, closed chain models incorporating measurement systems for calibration are less well understood. Difficulties arise from the issue of identifiability of parameters. This problem was studied in detail with the intent of producing a standardized approach which would eliminate the ambiguity often encountered. Due in part to the unlimited number and type of measurement systems available, no one independant method is possible. However, a systematic approach to the problem which alleviates most of the difficulties is proposed. Several case studies are presented which not only illustrate this approach, but emphasize some of the subtleties encountered in modelling measurement systems.

II. THEORY

A. CLOSED LOOP KINEMATIC CHAIN MODELING

1. General Coordinate System Transformations

A large class of manipulators can be thought of as a series of links connected by either rotary or prismatic joints. Typical kinematic models consist of fixed coordinate frames attached to each of the links and a set of transformation equations between these coordinate frames. This section will develop generalized coordinate frame transformations. The following sections will then address standardized transformations followed by a development of the kinematic model for the PUMA-560. The following conventions will be used throughout this document:

- Bold lower case letters will refer to vectors. A preceding superscript refers to the frame the vector is associated with. A subscript identifies the frame in which the coordinates of the vector are referenced.
- Bold upper case letters will refer to matrices.
- Upper case letters, excluding F, correspond to points. Preceding superscripts and subscripts have an identical meaning as defined for vectors.
- Coordinate frames will be denoted by F^i where the superscript refers to an assigned number or designation.
- Double subscripted lower case letters will usually refer to a vector or point of the same letter. The first subscript refers to the component of the vector or coordinate of the point and the second subscript refers to the frame to which it is referenced.

Consider the coincident coordinate frames of Figure 1 in which the y and z axis have been rotated an angle ψ about the x axis. First, the $\mathbf{i}_o$, $\mathbf{j}_o$ and $\mathbf{k}_o$ unit vectors in the nonrotated frame will be described with respect to the rotated coordinate frame unit vectors $\mathbf{i}_1$, $\mathbf{j}_1$ and $\mathbf{k}_1$. This will then provide a method of describing the coordinates of point P, with coordinates p_{xo}, p_{yo} and p_{zo}, given the coordinates of point P, p_{x1}, p_{y1} and p_{z1}, with respect to the rotated axis. Clearly, rotation about the x axis does not alter the $\mathbf{i}_o$ unit vector or the x component of P. Consequently, this problem can be reduced to a planar analysis by projection onto the y-z plane.

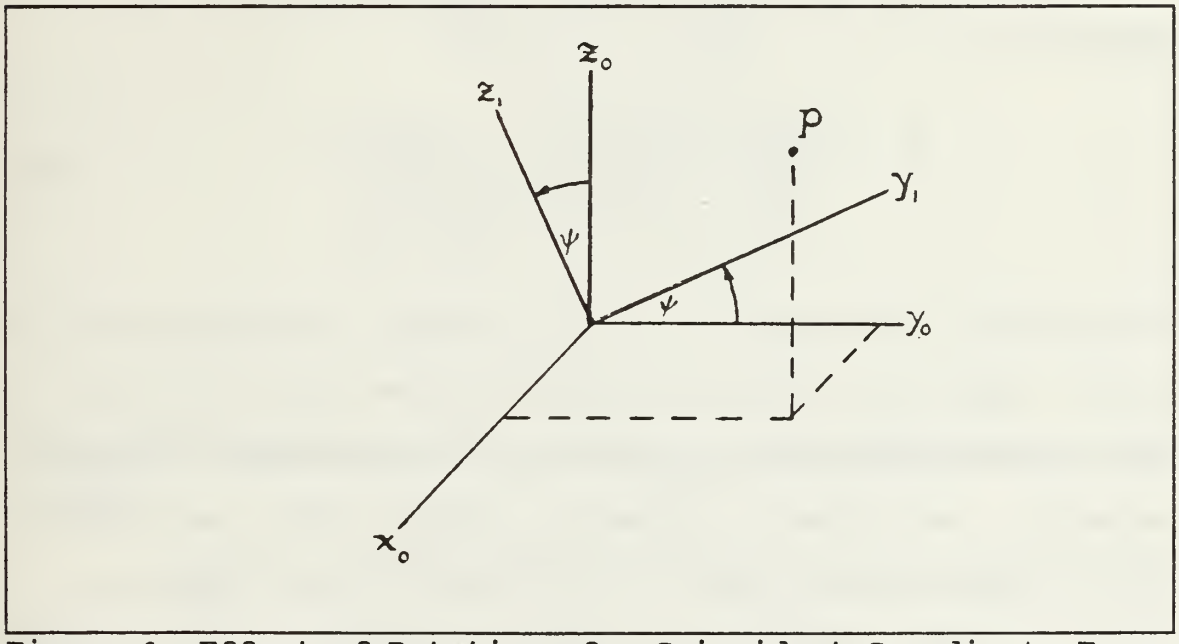

Figure 1. Effect of Rotation of a Coincident Coordinate Frame About the x Axis

Figure 2 illustrates this projection. Recalling that $\mathbf{i}_o$ and $\mathbf{i}_1$ are identical and noting the geometry of Figure 2, $\mathbf{i}_o$, $\mathbf{j}_o$ and $\mathbf{k}_o$ in terms of $\mathbf{i}_1$, $\mathbf{j}_1$ and $\mathbf{k}_1$ are given by Equation 1.

$$\begin{aligned} \boldsymbol{i}_0 &= \boldsymbol{i}_1 \\ \boldsymbol{j}_0 &= \cos\psi \boldsymbol{j}_1 - \sin\psi \boldsymbol{k}_1 \\ \boldsymbol{k}_0 &= \sin\psi \boldsymbol{j}_1 + \cos\psi \boldsymbol{k}_1 \end{aligned} \tag{1}$$

Rewriting Equation 1 in matrix form results in the following expression.

$$\begin{bmatrix} \boldsymbol{i} \\ \boldsymbol{j} \\ \boldsymbol{k} \end{bmatrix}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \boldsymbol{i} \\ \boldsymbol{j} \\ \boldsymbol{k} \end{bmatrix}_1 \tag{2}$$

Equation 2 can now be used to transform the coordinates of point P with respect to the rotated axis into coordinates in the nonrotated coordinate system by substituting the coordinates of point P for the unit vectors as shown in Equation 3.

$$\begin{bmatrix} p_{x0} \\ p_{y0} \\ p_{z0} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} p_{x1} \\ p_{y1} \\ p_{z1} \end{bmatrix} \tag{3}$$

Summarizing, the above 3x3 matrix can be interpreted in two ways. First, the matrix columns describe the orientation of the rotated frame with respect to the nonrotated frame. For example, the column 2 elements indicate that the rotated y axis, which is described by unit vector $\mathbf{j}_1$, has direction

$$\boldsymbol{j}_1 = 0\boldsymbol{i}_0 + \cos\psi \boldsymbol{j}_0 + \sin\psi \boldsymbol{k}_0 \tag{4}$$

Rewriting Equation 4,

$$\boldsymbol{j}_0 = 0\boldsymbol{i}_1 + \cos\psi \boldsymbol{j}_1 + \cos(90-\psi) \boldsymbol{k}_1 \tag{5}$$

and noting from Figure 2 that the angle $90-\psi$ is the angle

between the z_o and y_1 axis, then it can be seen that the column components are the familiar direction cosines. Secondly, the matrix can be thought of as a coordinate transformation matrix in which coordinates of points in one frame can be "transformed" into coordinates in a second frame as illustrated in Equation 3. These two interpretations of the 3x3 matrix transformation matrix will hold for all transformation matrices to follow.

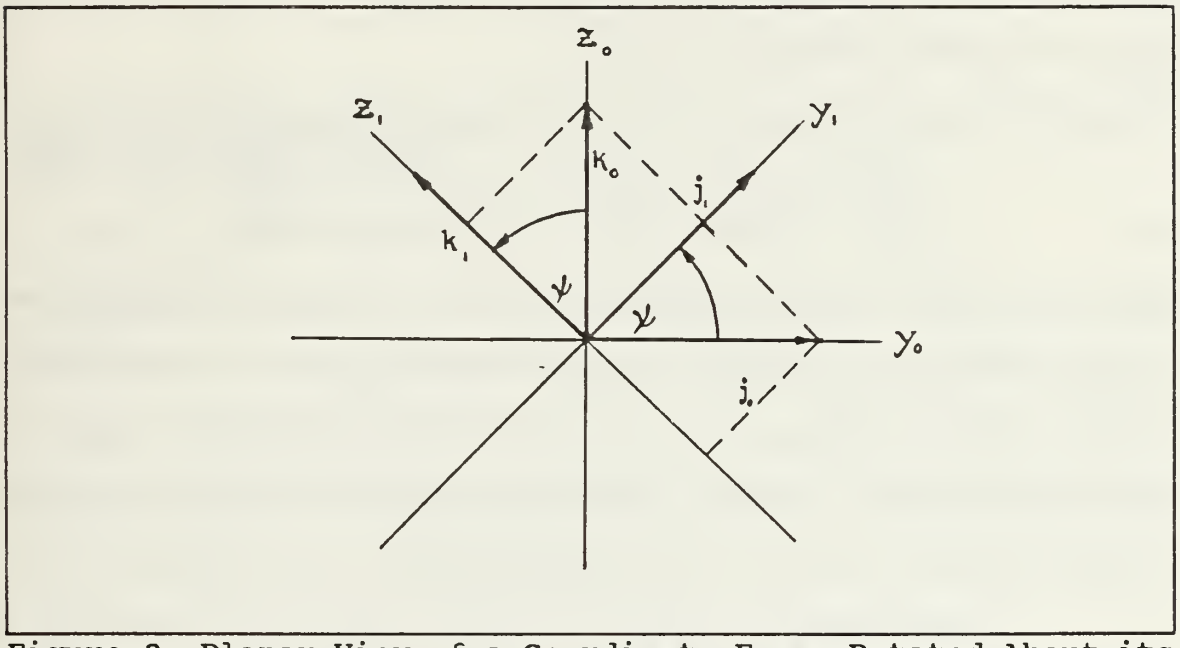

Figure 2. Planar View of a Coordinate Frame Rotated About its X Axis

If the rotated coordinate frame is now rotated about the y_1 axis an angle θ, then in a similar manner to the preceding analysis, the problem reduces to analysis in the x_1-z_1 plane. Referring to Figure 3, the i_1, j_1 and k_1 unit vectors in terms of the rotated axis x_2, y_2 and z_2 are given by Equation 6. These equations are again rewritten in matrix form as given in Equation 7.

$$\begin{aligned} \boldsymbol{i}_1 &= \cos\theta \boldsymbol{i}_2 + \sin\theta \boldsymbol{k}_2 \\ \boldsymbol{j}_1 &= \boldsymbol{j}_2 \\ \boldsymbol{k}_1 &= -\sin\theta \boldsymbol{i}_2 + \cos\theta \boldsymbol{k}_2 \end{aligned} \tag{6}$$

$$\begin{bmatrix} \boldsymbol{i} \\ \boldsymbol{j} \\ \boldsymbol{k} \end{bmatrix}_1 = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \boldsymbol{i} \\ \boldsymbol{j} \\ \boldsymbol{k} \end{bmatrix}_2 \tag{7}$$

Pre-multiplying both sides of Equation 7 by the matrix in Equation 2 as shown in Equation 8, will result in a transformation between the original coordinate frame and the twice rotated coordinate frame.

$$\begin{bmatrix} \boldsymbol{i} \\ \boldsymbol{j} \\ \boldsymbol{k} \end{bmatrix}_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \boldsymbol{i} \\ \boldsymbol{j} \\ \boldsymbol{k} \end{bmatrix}_2 \tag{8}$$

The preceding analysis demonstrates a valuable property of transformation matrices. Further rotations of a coordinate frame can be referenced to the original coordinate frame by post multiplying any previous transformation matrices by the transformation matrix that describes the next rotation.

A development similar to that leading to Equations 2 and 7 for a rotation about the z axis by angle ϕ results in the following transformation matrix R_z.

$$R_z = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

In addition to rotation of a coordinate frame about its axis, translation of the frame must be accounted for. Consider the coordinate frame with origin at point P in Figure

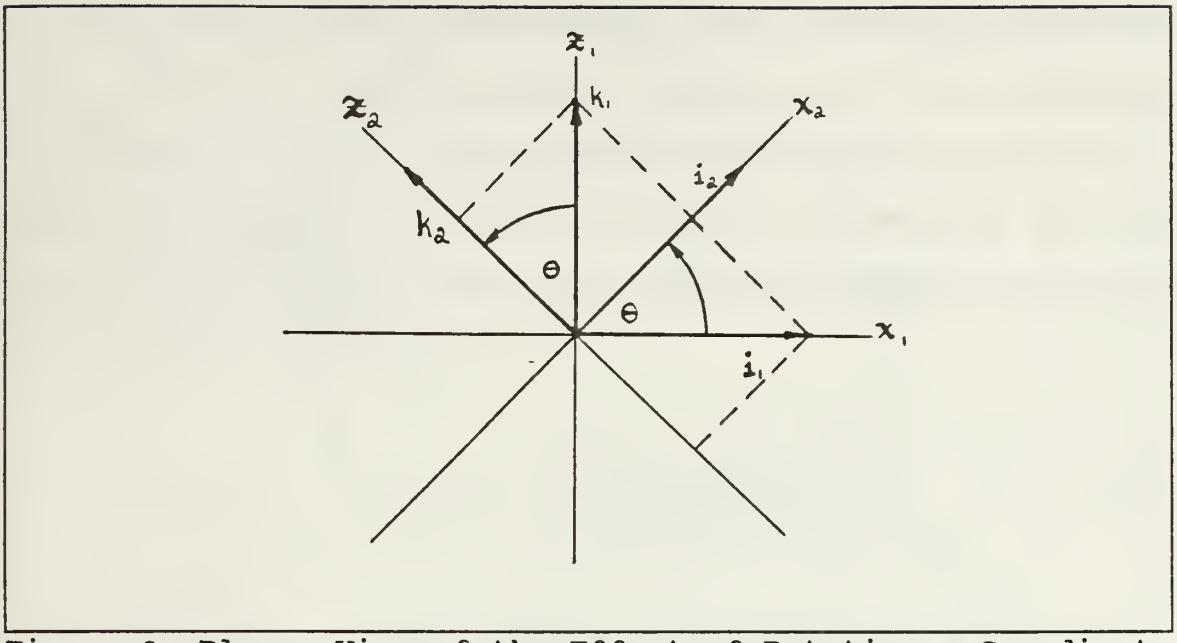

Figure 3. Planar View of the Effect of Rotating a Coordinate Frame About its Y Axis

4. Let R be the transformation matrix with elements r_{ij} that corresponds to rotation of a coordinate frame with origin at P and axis originally aligned with the coordinate frame at O. A point Q with coordinates q_{x1}, q_{y1} and q_{z1} with respect to the rotated coordinate frame can be expressed in coordinates with respect to point O by Equation 10.

$$\begin{bmatrix} q_{xo} \\ q_{yo} \\ q_{zo} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} q_{x1} \\ q_{y1} \\ q_{z1} \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \qquad (10)$$

It would be convenient if this translation transformation were incorporated in matrix form. This can be accomplished by forming an augmented 4x4 matrix and augmented vectors as illustrated in Equation 11. Matrices of this form are typically referred to as homogeneous transformation matrices.

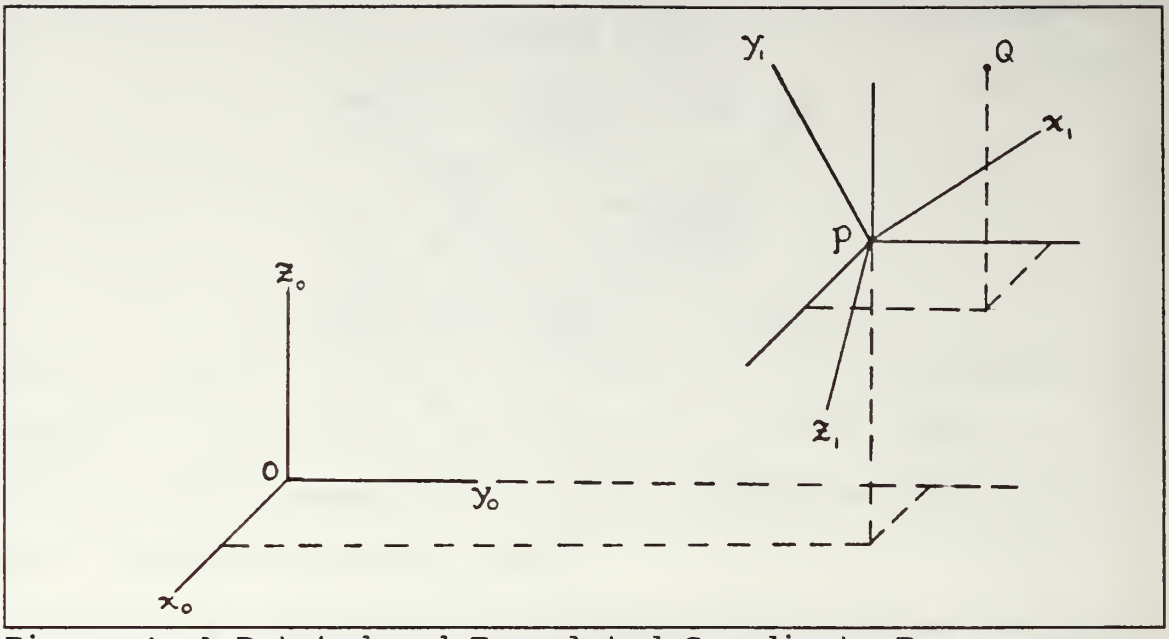

Figure 4. A Rotated and Translated Coordinate Frame

$$\begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \qquad (11)$$

The upper left 3x3 submatrix has the same orientation interpretation as noted earlier where the n, o and a elements are the direction cosines for the rotated x, y and z axis with respect to the original axis respectively. The upper 3 elements of column four of the matrix define the origin of the rotated and translated coordinate frame with respect to the original frame. With this interpretation in mind, the convention T_i^j will be used when referring to a transformation from frame i to frame j. When no subscript is indicated, T^j, then the transform is interpreted as being from a known reference frame, which will be clear from the context, to the

j^{th} coordinate frame. Additionally, the previously described 3x3 rotational transformation matrices can be expressed individually as 4 by 4 homogeneous transformation matrices as well. The convention Rot(x,ψ), Rot(y,θ) and Rot(z,ϕ) will be used in the following discussion and these matrices are shown in Equations 12 through 14.

$$Rot(x,\psi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

$$Rot(y,\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$Rot(z,\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

Additionally, a standard translation transformation will be denoted by Trans(x,y,z) and has the form shown in Equation 15 where the upper 3x3 rotational submatrix has been replaced with a 3x3 identity matrix. If the coordinate frame is only translated in one direction, say x, then the symbol Trans(x) will be used to denote this transformation and the p_y and p_z terms in Trans(x,y,z) are set to zero.

Given two coordinate frames F^0 and F^1, the homogeneous transformation matrix $\mathbf{T}_0^1$ and $\mathbf{x}_1$ in F^1, then $\mathbf{x}_0$ can be found by Equation 16.

$$Trans(x,y,z) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$\boldsymbol{x}_0 = \boldsymbol{T}_0^1 \, \boldsymbol{x}_1 \tag{16}$$

If $\mathbf{x}_0$ is known and $\mathbf{x}_1$ desired, then

$$\begin{aligned} (\boldsymbol{T}_0^1)^{-1} \, \boldsymbol{x}_0 &= (\boldsymbol{T}_0^1)^{-1} \, \boldsymbol{t}_0^1 \, \boldsymbol{x}_1 \\ (\boldsymbol{T}_0^1)^{-1} \, \boldsymbol{x}_0 &= \boldsymbol{x}_1 \\ \boldsymbol{T}_1^0 \, \boldsymbol{x}_0 &= \boldsymbol{x}_1 \end{aligned} \tag{17}$$

This of course requires knowledge of the inverse transformation matrix. However, as described by Paul [4], if a transformation matrix **T** has the elements of Equation 18 then its inverse is evaluated by Equation 19 where **n**, **o**, **a** and **p** are the four column vectors of **T** and "·" is the usual vector dot product operator.

$$\boldsymbol{T} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

$$\boldsymbol{T}^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\boldsymbol{p}\cdot\boldsymbol{n} \\ o_x & o_y & o_z & -\boldsymbol{p}\cdot\boldsymbol{o} \\ o_x & o_y & o_z & -\boldsymbol{p}\cdot\boldsymbol{a} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

As noted earlier, given transformations $\mathbf{T}_0^1$ and $\mathbf{T}_1^2$, then the transformation from F^0 to F^2 can be calculated by post

multiplying T_0^1 by T_1^2. This product can then be designated as T_0^2 or simply as T^2 where the reference frame F° is assumed. Figure 5 illustrates various transformations by use of a directed graph. Each node represents a coodinate frame and the directed paths represent transformations in the given direction.

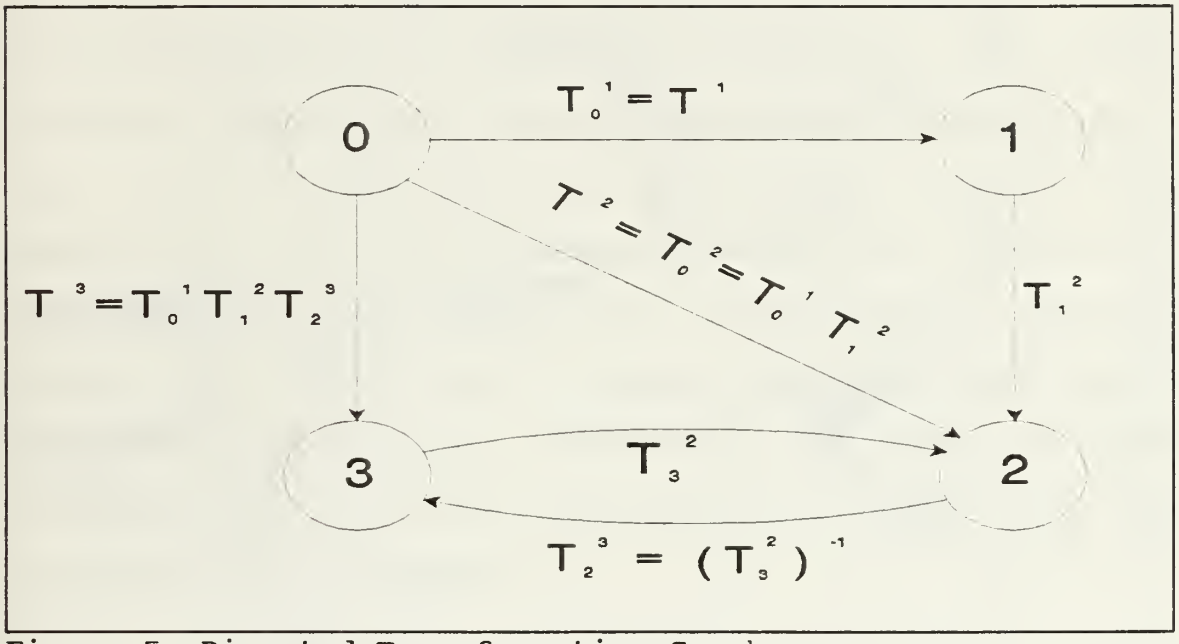

Figure 5. Directed Transformation Graph

2. Roll, Pitch, Yaw and Translation Transformations

To transform between two coordinate frames fixed in space as shown in Figure 6 requires, in general, 3 rotations and 3 translations. Noting that the order in which the transformations occur is important, adoption of one of the standards will help avoid confusion. The standard used for this type of orientation transformation in this work is roll, pitch and yaw.

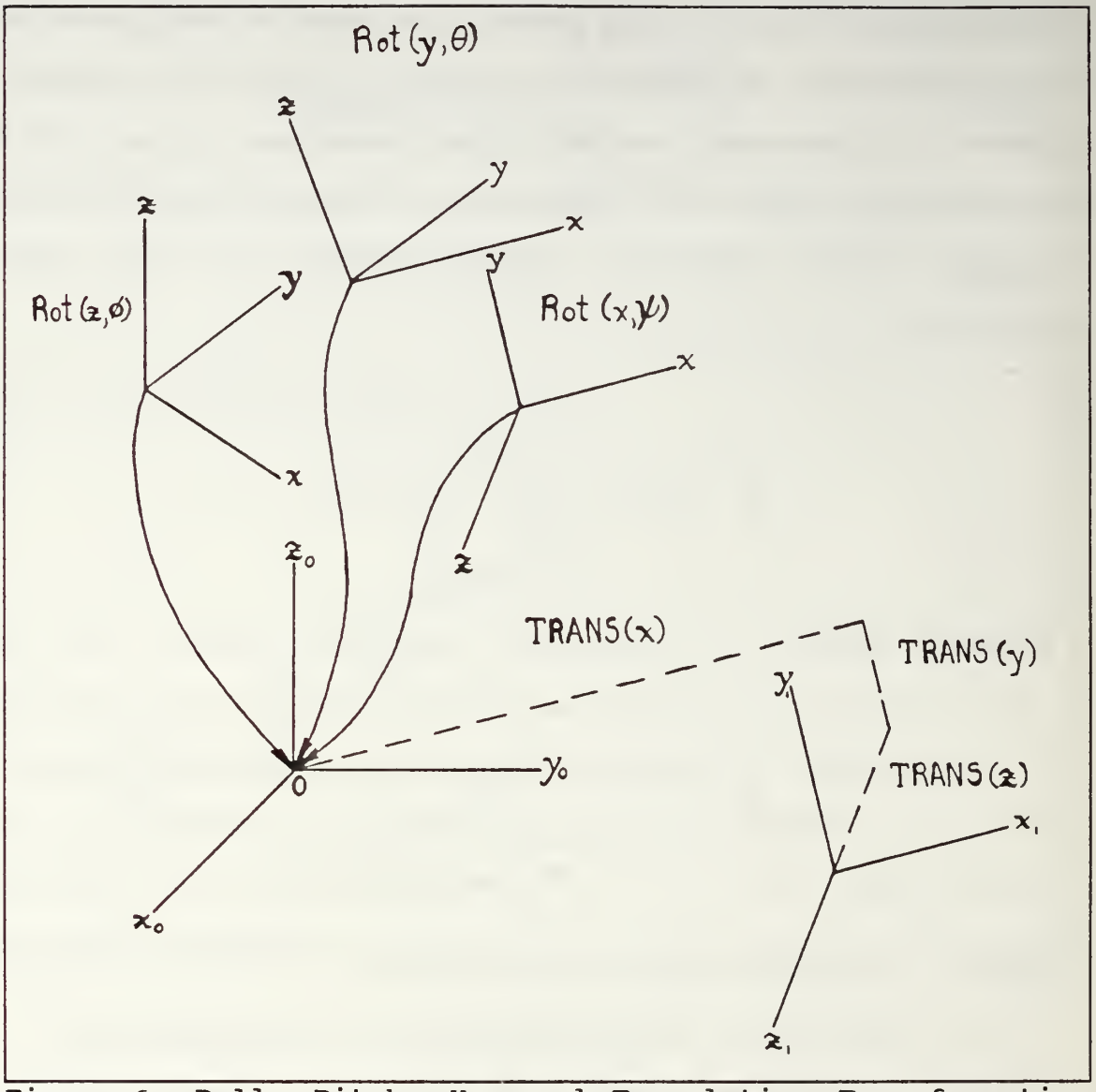

Figure 6. Roll, Pitch, Yaw and Translation Transformation Between Two Fixed Coordinate Frames

The roll, pitch and yaw rotation transformation will be denoted as RPY(ϕ,θ,ψ) and is a product of the previously described rotation matrices as follows

$$RPY(\phi,\theta,\psi) = Rot(\phi,z)\ Rot(\theta,y)\ Rot(\psi,x) \qquad (20)$$

Carrying out the indicated matrix multiplication, the elements of RPY(ϕ,θ,ψ) are given in Equation 21 where cosine and sine are denoted by c and s for brevity.

$$RPY(\phi,\theta,\psi) = \begin{bmatrix} c\phi c\theta & c\phi s\theta s\psi - s\phi c\psi & c\phi s\theta c\psi + s\phi s\psi & 0 \\ s\phi c\theta & s\phi s\theta s\psi + c\phi c\psi & s\phi s\theta c\psi - c\phi s\psi & 0 \\ -s\theta & c\theta s\psi & c\theta c\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (21)$$

With the orientation now specified by RPY(ϕ,θ,ψ), it is only necessary to specify the translations between the two frames. This can be accomplished by multiplying RPY(ϕ,θ,ψ) by Trans(x,y,z). Note that premultiplication by Trans(x,y,z) implies that the translations occur on with respect to F^o axis and that post multiplication implies that the translations occur with respect to the rotated axis. The convention used throughout this work is post multiplication. Summarizing, the transform T_o^1 is calculated by

$$\begin{aligned} T_0^1 &= RPY(\phi,\theta,\psi)\, Trans(x,y,z) \\ &= Rot(\phi,z)\, Rot(\theta,y)\, Rot(\psi,z)\, Trans(x,y,z) \end{aligned} \quad (22)$$

with each step in the transformation illustrated graphically in Figure 6. Transformations of this form will be denoted by RPYT(ϕ,θ,ψ,x,y,z) or simply RPYT.

3. Denavit-Hartenburg Transformations

As indicated in the preceding section, 3 axis rotations and 3 translations are, in general, required to transform between two coordinate frames. However, the geometry of successive links of a manipulator imposes constraints on the transformation between coordinate frames fixed in these

links and a subsequent reduction in the number of rotations and/or translations required. The form of these transformations is a function of the link geometry, the type of interconnecting joints and the placement (orientation and position) of the link frames. Clearly a systematic approach to frame allocation is desirable if not essential. One widely accepted systematic approach is the Denavit-Hartenburg method.

Typical manipulators, such as the PUMA-560 illustrated in Figure 7, consist of a series of links and joints. An n degree of freedom manipulator will have n links and n joints. The links and joints of the manipulator are labeled in the following manner. The first joint is labeled 1 and the joint number is incremented by one for each successive joint. Link i lies between joint i and joint i+1. The base or base link is defined as link 0.

Figure 8 illustrates a generic link. The parameter a_n is the common normal distance between joint axis n and n+1 and is usually referred to as the link length. A plane, normal to a_n at the intersection of the common normal a_n and joint axis n+1 will by definition contain joint axis n+1 and lines parallel to joint axis n. The angle between joint axis n+1 and a line parallel to joint axis n in the plane is designated α_n, and is generally referred to as the link twist angle. In addition to a_n, common normal a_{n-1} intersects joint axis n. The distance between these two common normals along the axis is designated d_n and is usually referred to as the distance

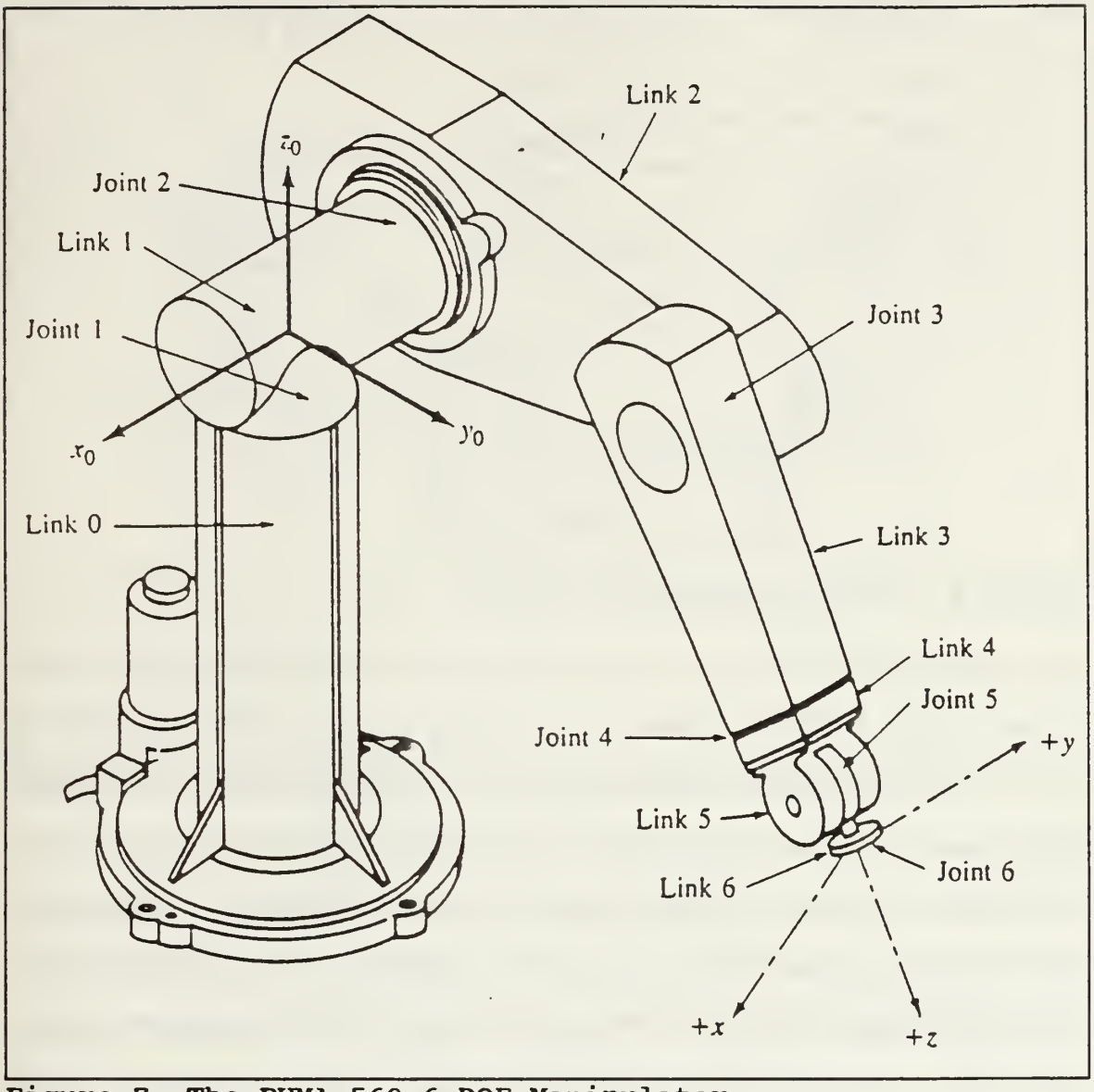

Figure 7. The PUMA 560 6 DOF Manipulator

between the links or the offset distance. Similarly, d_{n+1} is the offset distance along joint axis n+1 between common normals a_n and a_{n+1}. A fourth parameter θ_n, is defined as the angle in a plane perpendicular to joint axis n between common normals a_{n-1} and a_n.

With these parameters identified, assignment of coordinate frames to each link based on these parameters can

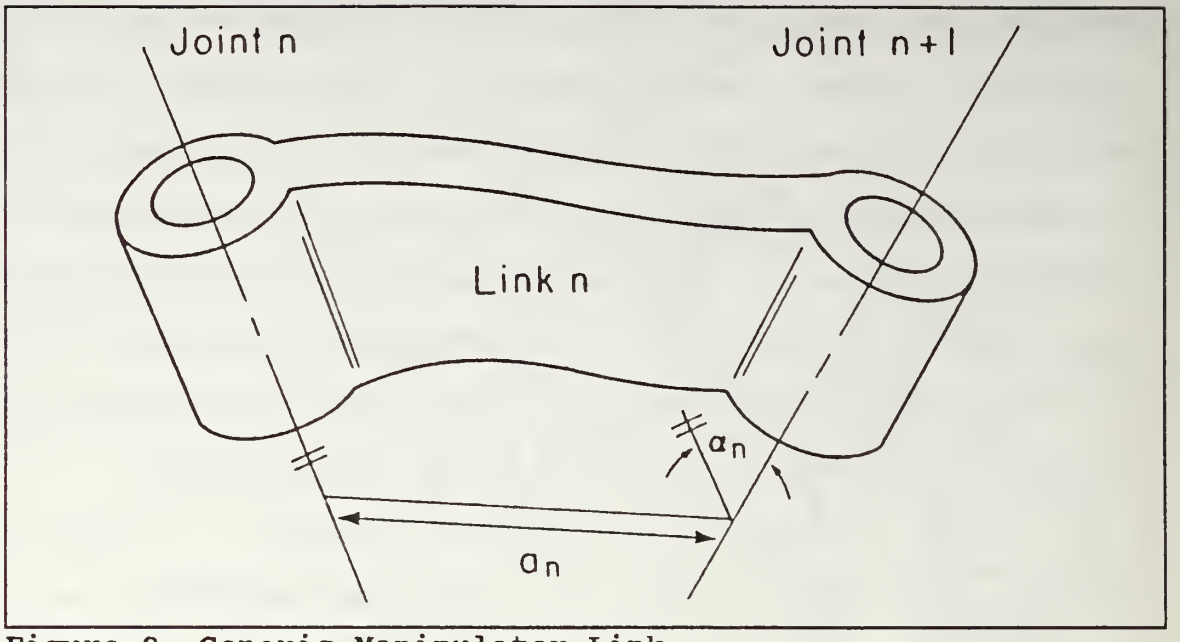

Figure 8. Generic Manipulator Link

be accomplished. Figure 9 illustrates the process for the case of revolute joints. Frame n, F^n, lies at the intersection of a_n, the common normal between joint axis n and n+1, and joint axis n+1. If the joint axis intersect, then the intersection is chosen as the origin which is consistent with the above description noting that a_n is zero. If the axes are parallel, then the frame origin is chosen so that the offset distance is zero for the next defined frame origin. The coordinate frame axis are aligned as follows. z_n lies on joint axis n+1. The x_n axis is aligned with a_n when it exists. If a_n does not exist, as in the case of intersecting joint axis, x_n is aligned perpendicular to joint axis n and n+1. The zero position of the joint variable, θ_n, is defined when x_{n-1} and x_n are both parallel and in the same direction.

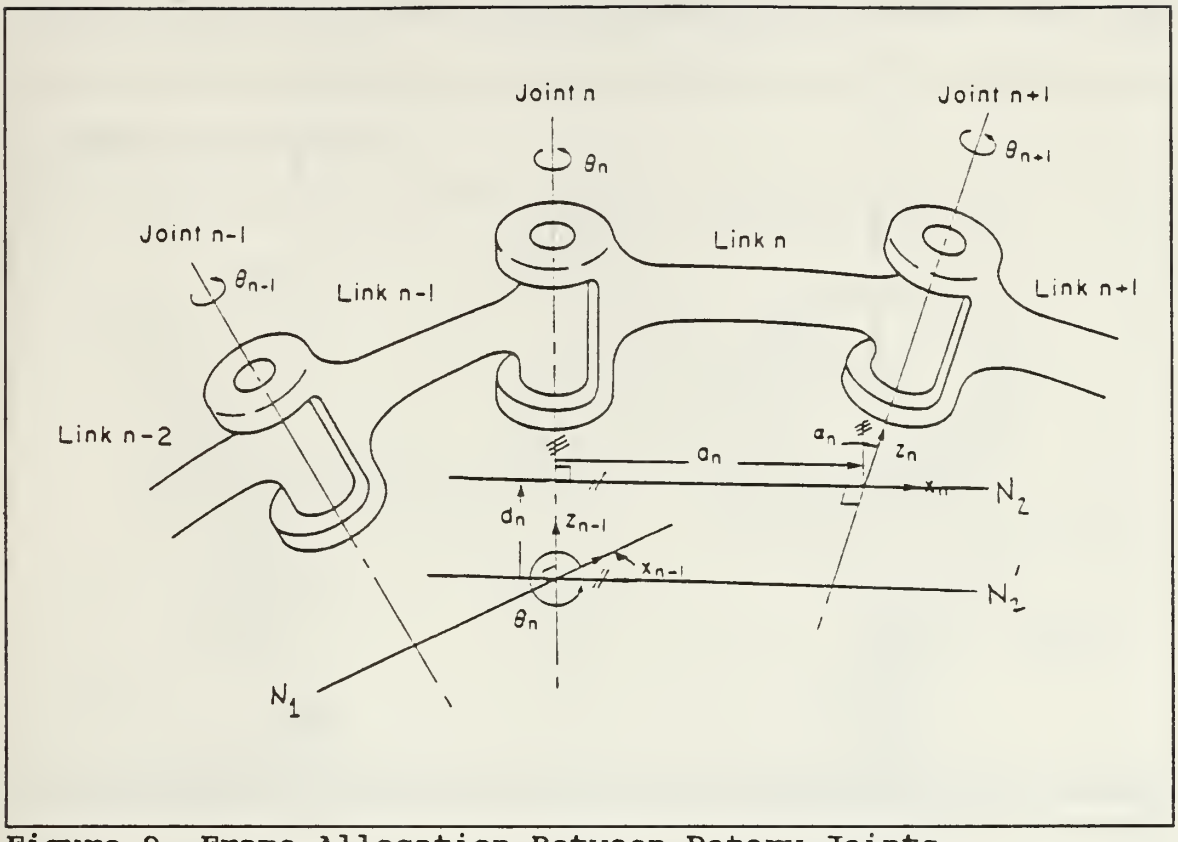

Figure 9. Frame Allocation Between Rotary Joints

Before proceeding with an explanation of the method of assigning coordinate frames for prismatic joints, a simple illustration of a fundamental difference between prismatic and rotary joints is offered. Consider a point P traveling in a circular path in space as illustrated in Figure 10a. The path of P defines a plane and hence a perpendicular direction. Furthermore, the center of the circle in the above plane clearly defines a point in space with which to reference point P. On the other hand, linear motion of P as shown in Figure 10b offers no such reference point. In fact, the axis is indistinguishable from any other parallel axis.

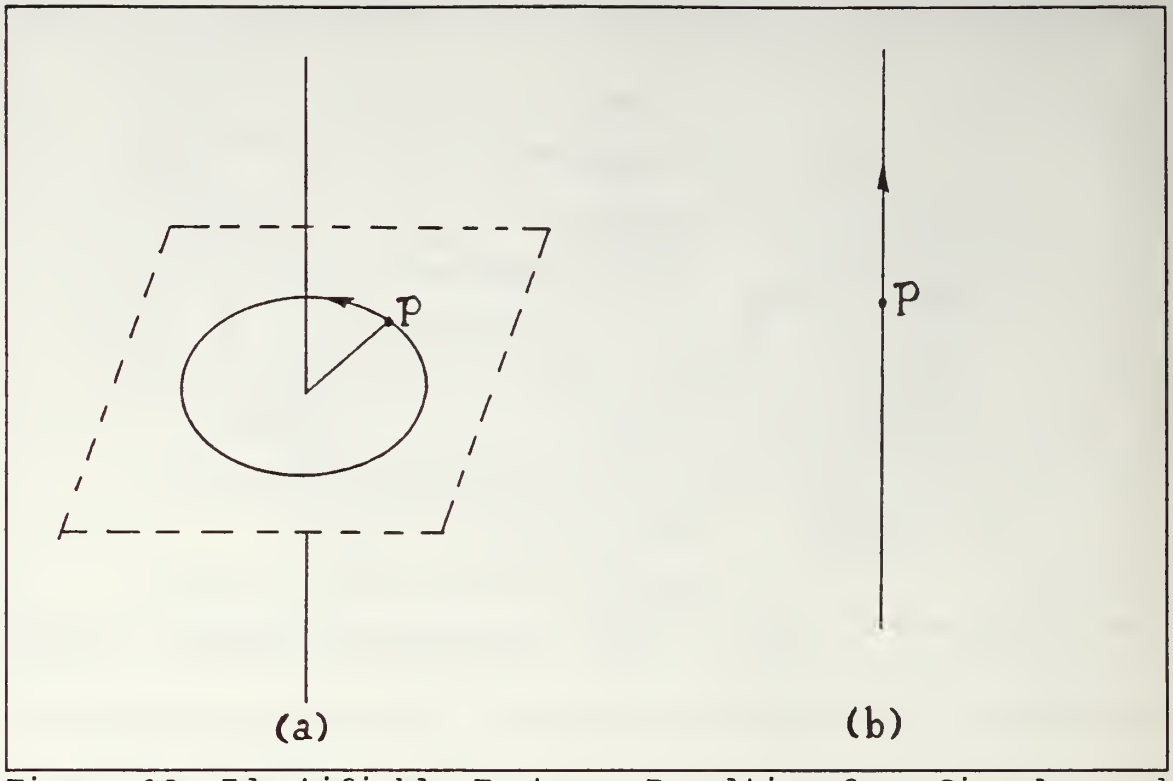

Figure 10. Identifiable Features Resulting from Circular and Linear Motion

Figure 11 illustrates a prismatic joint located between two rotary joints. For the prismatic joint, the joint distance d_n is the joint variable. As noted in the preceding paragraph, the position of the joint axis is undefined and only the direction of the axis is known. Consequently, the common normal parameter a_n is meaningless. With this in mind, the origin of F^n is placed coincident with the next defined link coordinate frame origin. Note that this placement may be ambiguous if the prismatic joint were at or near the end of a serial link manipulator and an alternative placement may be necessary. When placed at the next defined origin, the z_n axis is aligned with joint axis n+1. The x_n axis is positioned

perpendicular to z_n and the prismatic joint axis n. The zero position for prismatic joint is defined when the distance d_n in Figure 11 is zero.

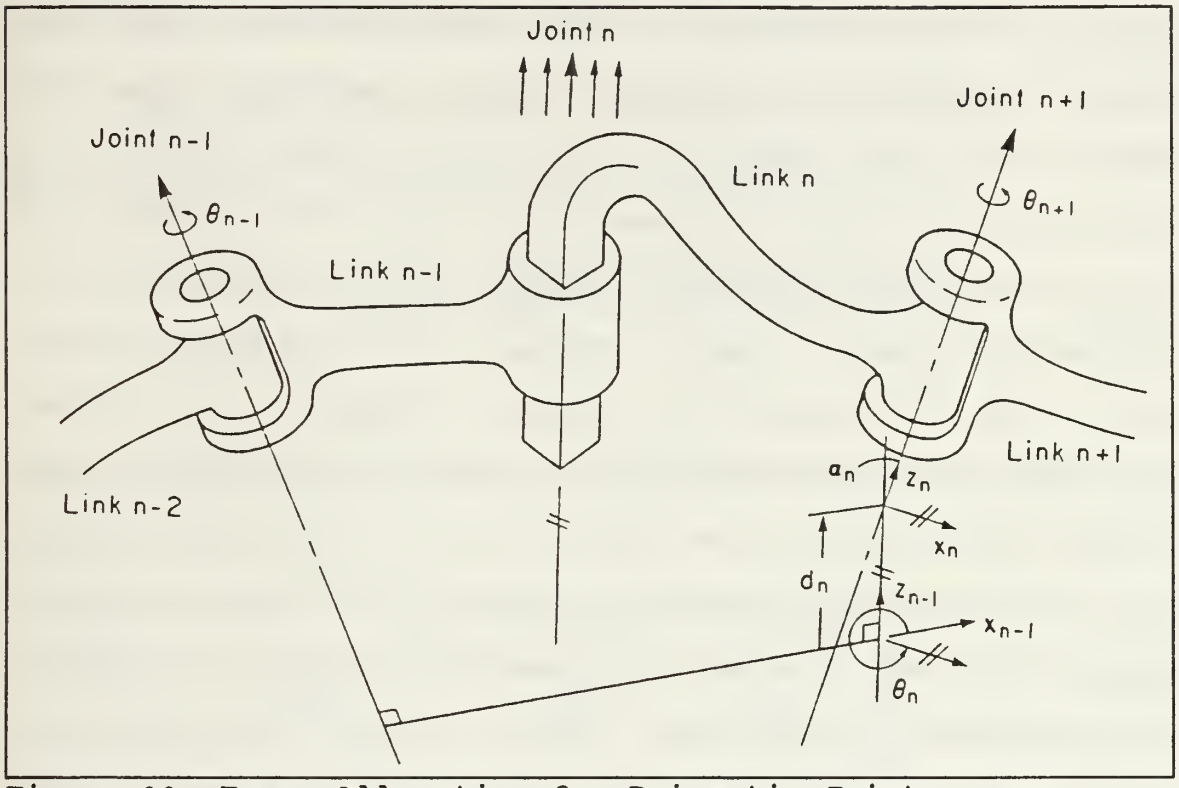

Figure 11. Frame Allocation for Prismatic Joints

With the manipulator placed in the zero position for both rotary and prismatic joints as defined in the preceding paragraphs, positive sense of rotations can be defined and then the appropriate sense of all the z axis determined. According to Paul [Ref. 5], the base link frame of the manipulator, F^0, will be coincident with the origin of F^1. However, such an allocation will not afford a standard transformation from F^0 to F^1 unless joint axis 1 and 2 happen to intersect. A more appropriate location of the base frame will be to place the z_0 axis coincident with joint axis 1 and

in a direction satisfying the right hand rule. The x_0 axis should be aligned with the common normal between joint axis 1 and joint axis 2. If joint axis 1 and 2 intersect, then x_0 should lie perpendicular to the two axes. Note that this method is generally consistent with the previously described method with the exception that d_0 is defined zero. This definition of F^0 is not necessary but other locations may lead to difficulty in determining identifiable parameters as will be discussed later. Finally, the end link coordinate frame, link 6 for a 6 degree of freedom manipulator, is placed coincident with the preceding frame and with the z axis aligned with the previous frame's z axis. For a given set of joint variables and parameters, the end link frame is fixed in space. With this in mind, a 6 degree of freedom homogeneous transformation, such as Roll, Pitch, Yaw and Translation, is then necessary to describe the pose of an end effector not coincident with the end link frame.

Now that the pose of each link frame is defined, the transformation matrices between frames can be developed. The type and order of the rotation and translation transformations which form the overall transformation from frame to frame follows in a natural way from the path from F^{n-1} to F^n. Referring back to Figure 9, F^{n-1} is rotated θ_n about z_{n-1} so that rotated x_{n-1} is aligned in the direction of a_n. The rotated frame is then translated d_n in the z_{n-1} direction followed by a translation a_n in the rotated x_{n-1} direction. The rotated and

twice translated frame now only requires alignment of its z axis with joint axis n+1 which is accomplished by a rotation about the rotated x_{n-1} axis, which now is equivalent to x_n, an angle α_n. In equation form, T_{n-1}^{n} can be expressed by

$$T_{n-1}^{n} = Rot(z,\theta)\,Trans(0,0,d)\,Trans(a,0,0)\,Rot(x,\alpha) \qquad (23)$$

Carrying out the indicated matrix multiplication,

$$T_{n-1}^{n} = \begin{bmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & a\cos\theta \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & a\sin\theta \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (24)$$

For a prismatic joint, T_{n-1}^{n} reduces to the matrix in Equation 25.

$$T_{n-1}^{n} = \begin{bmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & 0 \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (25)$$

4. Modified Denavit-Hartenburg Transformations

In general, application of the Denavit-Hartenburg method will result in an accurate model of a manipulator. However, some limitations exist such as the previously noted potential ambiguity with regard to prismatic joints and a disproportionate model [Ref. 6].

A proportional model can be defined as one in which changes in any model parameter will result in changes in other model parameters of the same order of magnitude. This is clearly not the case with Denavit-Hartenburg when modelling parallel or nearly parallel consecutive joints. Consider the

two nearly parallel intersecting axis lying in a plane P in Figure 12. A small rotation of axis uu' in the plane P will result in a large change in the location of the point of intersection. Large changes in the location of the common normal a_n between nearly parallel axis can occur in much the same way. Furthermore, small variations in orientation between nearly parallel axis may actually place them parallel in which case a unique common normal no longer exists. These potentially large variations in the location of the common normal corresponds to equally large changes in the parameter d_n in the Denavit-Hartenburg method. It should be noted that this apparent flaw in the Denavit-Hartenburg method is only an issue when developing a kinematic model for calibration where the fixed model parameters become variables and are perturbed numerically. Disproportionate changes will frequently result in numerical instability, an issue which will be addressed in greater detail in following sections. A model developed for the sole purpose of determining the end effector pose with respect to a given reference frame, the so called forward kinematic solution, will have, excluding the joint variables, fixed values based on assumed geometry and therefore immune to problems of proportionality. Of course the validity of the model is only as good as the geometric assumptions and this provides the motivation for calibration.

The following modification to the standard Denavit-Hartenburg transformations will result in a proportionate

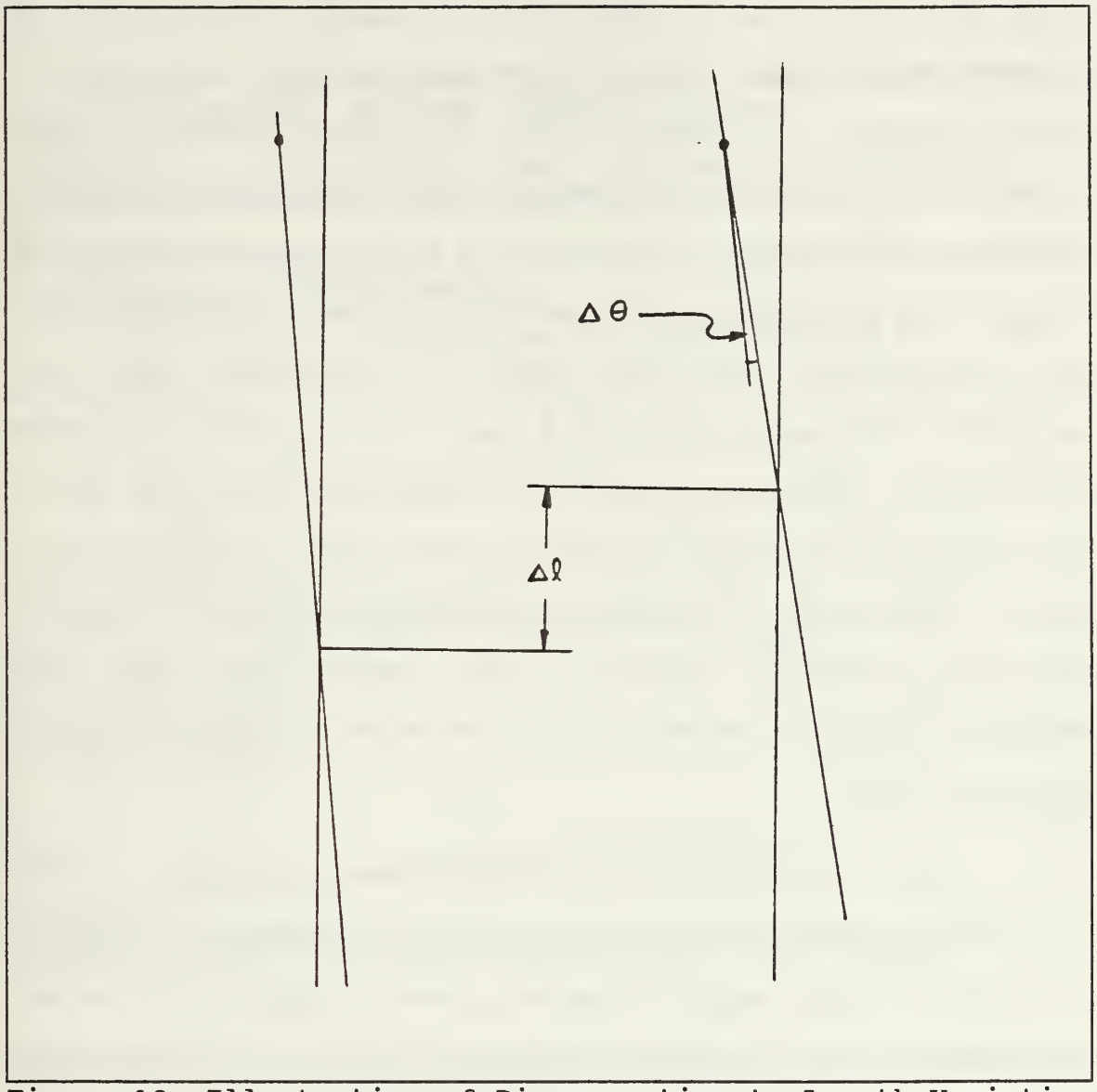

Figure 12. Illustration of Disproportionate Length Variation Due to Small Axis Rotation

model for consecutive revolute joints. This modification follows one proposed by Hayati and Mirmirani [Ref. 7]. Rather than specifying a common normal distance between two parallel or nearly parallel joint axis n-1 and n, define a plane that is perpendicular to joint axis n-1 and located at the origin of F^{n-1} as illustrated in Figure 13. The intersection of this

plane and joint axis n defines the origin of F^n which is defined whether the axis are parallel or not. As shown, a rotation about z_{n-1} an angle θ will place the rotated x_{n-1} axis on the line $O_{n-1}O_n$ and thus becomes the first transformation. Translation along $O_{n-1}O_n$ a distance r_n will place the origin of the rotated and translated coordinate frame coincident with O_{n-1}. In general, rotation about two different axes are required to align the third axis of a frame in some arbitrarily specified direction. Therefore, to align the z axis of the rotated and translated frame with joint axis n+1, which maintains continuity with the standard Denavit-Hartenburg method, rotations about the x and y axis are required. Equation 26 summarizes the above transformations in equation form.

$$T^n_{n-1} = Rot(z,\theta_n)\,Trans(r_n,0,0)\,Rot(x,\alpha_n)\,Rot(y,\beta_n) \qquad (26)$$

Carrying out the indicated matrix multiplication of Equation 26 results in the matrix elements given in Equation 27 where, for brevity, c and s have been substituted for sine and cosine respectively.

$$T^n_{n-1} = \begin{bmatrix} -s\alpha s\beta s\theta + c\beta c\theta & -c\alpha s\theta & s\alpha c\beta s\theta + s\beta c\theta & rc\theta \\ s\alpha s\beta c\theta + c\beta s\theta & c\alpha c\theta & -s\alpha c\beta c\theta + s\beta s\theta & rs\theta \\ -c\alpha s\beta & s\alpha & c\alpha c\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (27)$$

Comparing Equation 26 and Equation 23 it can be seen that a transformation **T** as in Equation 28 will satisfy any one of the three manipulator transformations. This is accomplished

by:

- setting ß equal to zero for standard Denavit-Hartenburg transformations between revolute joints;
- setting the parameters ß and a to zero for standard Denavit-Hartenbur transformations for prismatic joints;
- setting the parameter d to zero for the Modified Denavit-Hartenburg transformation;

$$T = Rot(z,\theta)\,Trans(z)\,Trans(x)\,Rot(x,\alpha)\,Rot(y,\beta) \qquad (28)$$

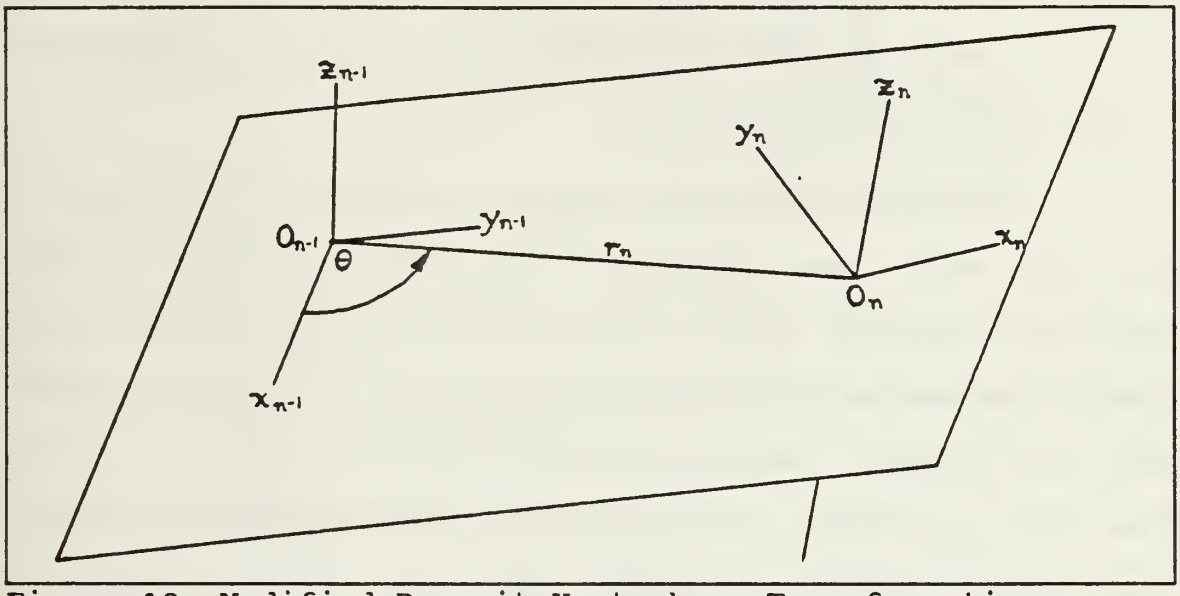

Figure 13. Modified Denavit-Hartenburg Transformation

This transform T is the standard transform used in this work for manipulator link to link transformations. The transformation allows for standardizing parameter information in tabular form as well as a single subroutine computer code for manipulator transformations. The elements of this transform are given in Equations 29 where t_{ij} is an element in the i^{th} row and j^{th} column of T. Any future reference to a transform between links of a manipulator will be considered to be in the form of Equation 29.

$$
\begin{aligned}
t_{11} &= \cos\theta\cos\beta - \sin\theta sin\alpha\sin\beta \\
t_{12} &= \sin\theta\cos\alpha \\
t_{13} &= cos theat\sin\beta + \sin\theta\sin\alpha\cos\beta \\
t_{14} &= a\cos\theta \\
t_{21} &= \sin\theta\cos\beta + \cos\theta\sin\alpha\sin\beta \\
t_{22} &= \cos\theta\cos\alpha j \\
t_{23} &= \sin\theta\sin\beta - \cos\theta\sin\alpha\cos\beta \\
t_{14} &= a\sin\theta \\
t_{31} &= -\cos\alpha\sin\beta \\
t_{32} &= \sin\alpha \\
t_{33} &= \cos\alpha c\beta \\
t_{34} &= d \\
t_{41} &= t_{42} = t_{43} = 0 \\
t_{44} &= 1
\end{aligned}
\qquad (29)
$$

5. Other Special Cases

Up to this point, two specific types of transformation matrices have been developed, the Roll, Pitch, Yaw and Translation matrix, RPYT, and the modified Denavit-Hartenburg transformation matrix. These matrices will be used exclusively for the kinematic models developed in this thesis. However, there are cases when transformations must be described between coordinate frames and other less well defined geometric quantities. For example, suppose a measurement system somehow clearly defines a point M in space but fails to define a set of axis. To incorporate this measurement system into the model of the manipulator, it is necessary to develop a transform from the point, M, to a frame in the manipulator, normally F°, as illustrated in Figure 14. There are no axes to align nor are there axes on which to translate from M to the origin of F°. However, a number of alternatives are available to transform from F° to M. Any set of three of the six variables of RPYT can be used to perform such a transformation so long

as one of the variables is a translation. A transformation satisfying the preceding, S_o^M, can be formed from RPYT by setting the irrelevant variables to zero. Such a transform fixes a frame at point M which has axis orientation dependant on both F^o and S_o^M. A transformation from M to F^o can then be calculated by inverting S_o^M. A frame defined in the manner above will be denoted with italicized print so as to distinguish it from an independently defined frame.

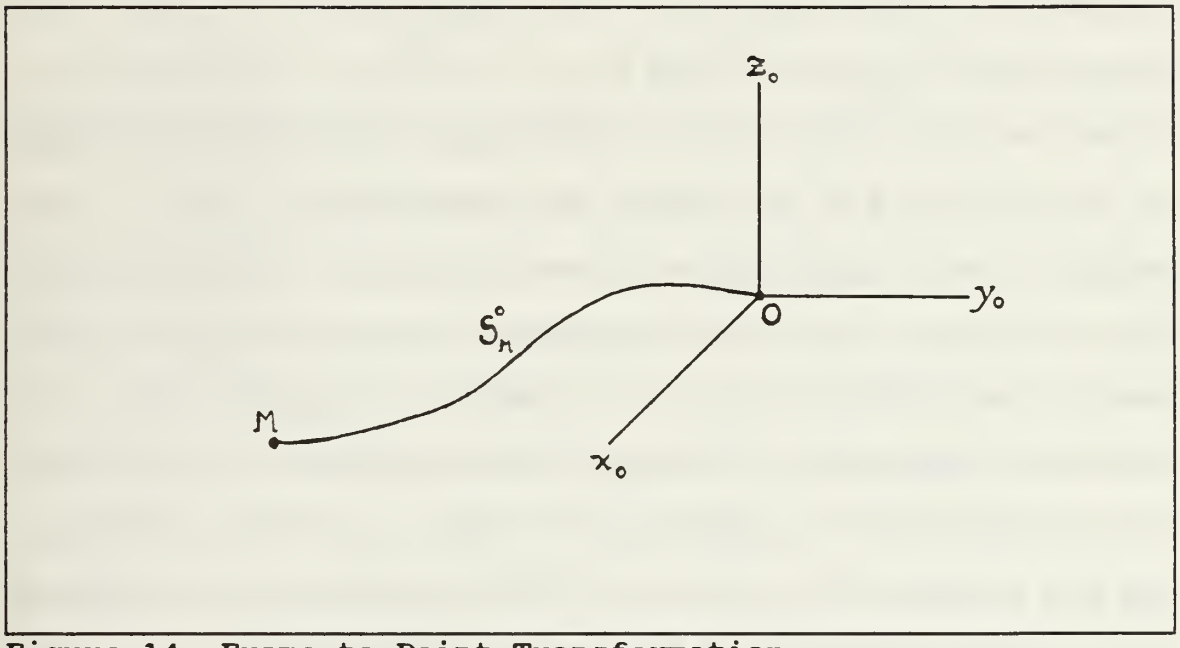

Figure 14. Frame to Point Transformation

6. The Kinematic Chain

The path through the series of frames illustrated in Figure 15 can be thought of as a kinematic chain where the transforms between frames are analogous to links. As described earlier, the pose of the last frame in the chain with respect to the origin of the chain can be formed by post multiplying each transform in sequence as shown. Before applying this

concept to a manipulator, it should be noted that it is frequently convenient to define an external reference frame in the manipulator workspace since F^{o}, as defined earlier, will normally be internal to the manipulator structure and not easily measured or referenced from the manipulator workspace. In the following experimental work, this external reference frame is coincident with the measurement system reference frame and is referred to as F^{m}. If F^{m} is a fully defined and independent frame then all six parameters of RPYT are necessary to transform from F^{m} to F^{o} and this transformation, T_{M}^{o} becomes the first link in the chain. The following links in the chain are described by appropriate forms of the modified Denavit-Hartenburg transformations. Recalling that the last frame in the manipulator in accordance with the Denavit-Hartenburg method is placed coincident with the previous frame, then a RPYT transform is required to transform from the last Denavit-Hartenburg frame to a frame located at the end effector, F^{E}. The pose at F^{E} with respect to F^{M} denoted T_{M}^{E} or simply T^{E} can be calculated by Equation 30.

$$T^{E} = T_{M}^{0}\, T_{0}^{1} \cdots T_{n-1}^{n}\, T_{n}^{E} \tag{30}$$

7. The Thirty Parameter Puma Kinematic Model

As noted by Mooring, Roth and Driels [Ref. 8], the number of parameters N in a complete model is

$$N = 4R + 2P + 6 \tag{31}$$

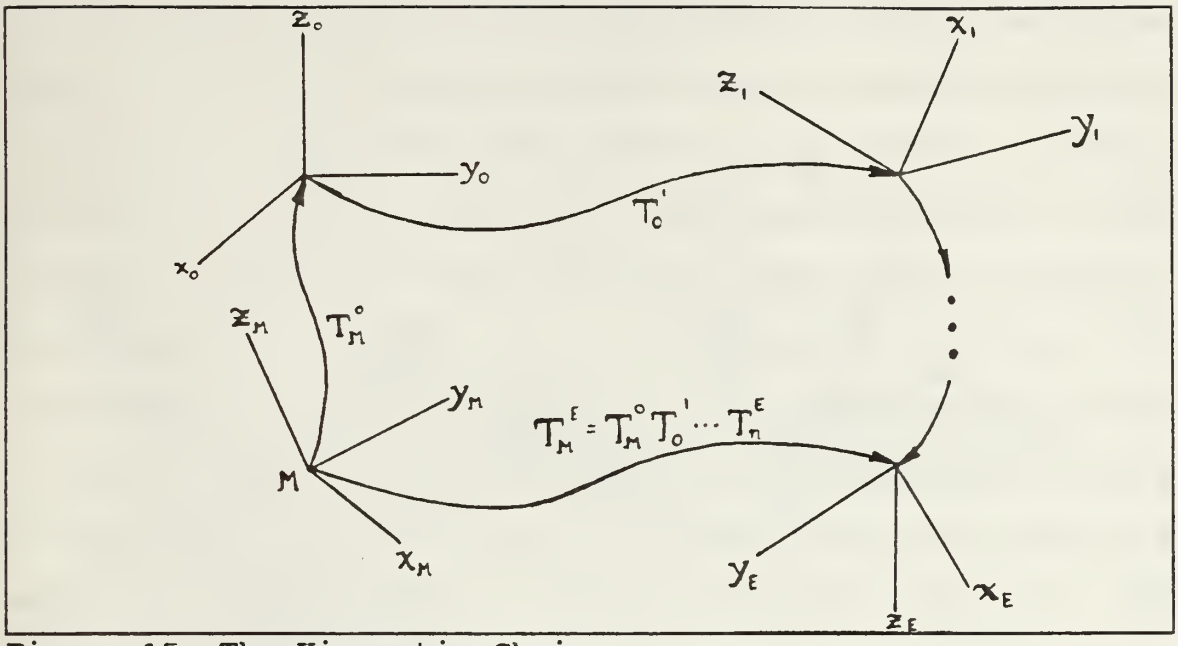

Figure 15. The Kinematic Chain

where R is the number of rotary joints and P is the number of prismatic joints. Complete in this sense means that the model contains a fully defined independent external reference frame and an independently defined tool frame. The PUMA-560 consists of 6 rotary joints which by Equation 31 suggests that the complete model will have 30 parameters. The PUMA-560 30 parameter model is developed in the following Section and the actual frame locations are illustrated in Figure 16.

The location of the external or measurement system reference frame is arbitrary within the manipulator's workspace. All six parameters of an RPYT transformation will be required to transform from F^M to F^0. At this point, it will be useful to distinguish between joint parameters and joint variables as used in calibration. The joint variable, denoted θ_i, is associated with the amount joint i is rotated from its

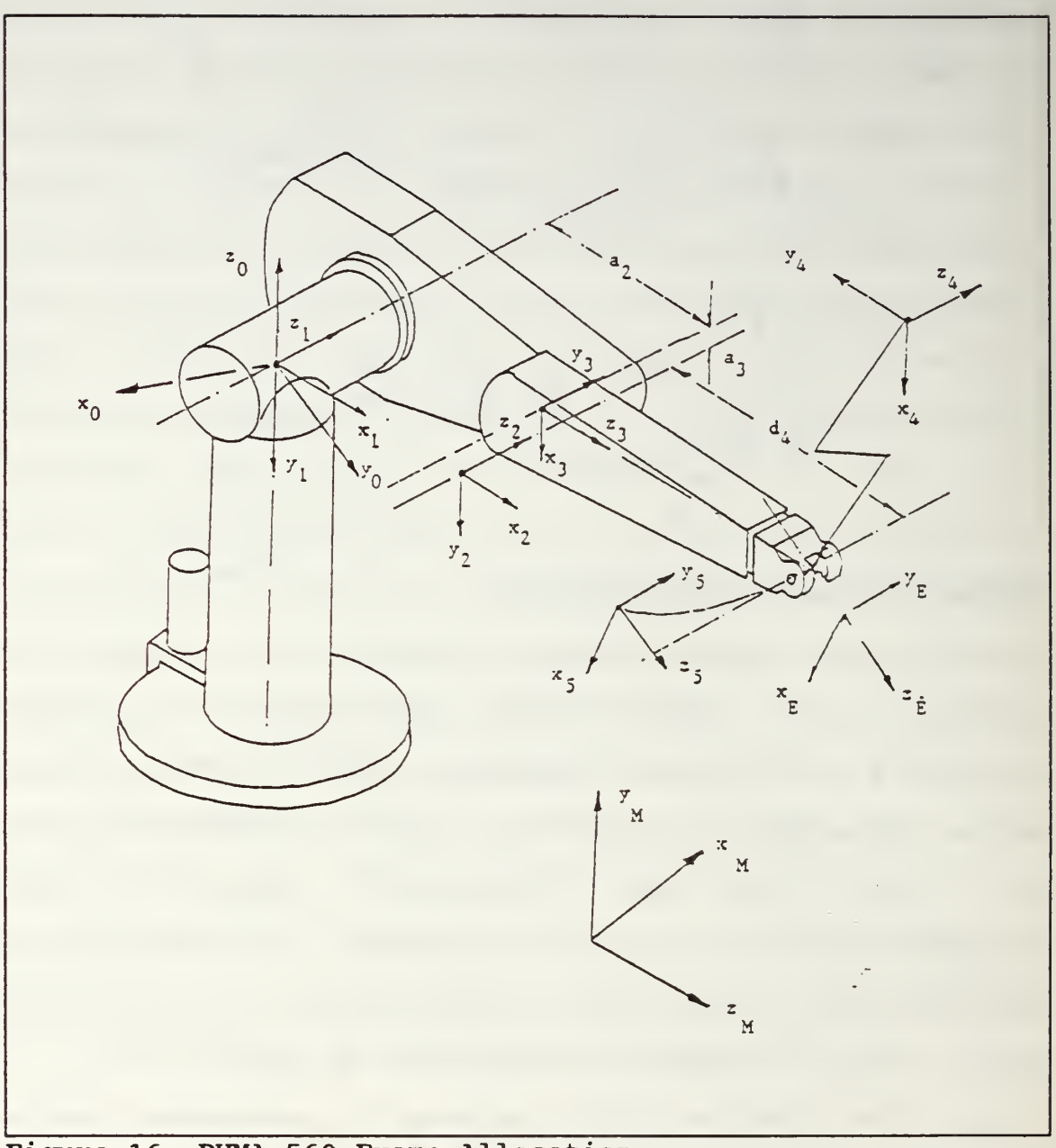

Figure 16. PUMA-560 Frame Allocation

zero position as defined by a joint encoder. The joint parameter, denoted $\delta\theta_i$ is essentially the error between the encoder zero and the actual zero position of the i^{th} joint as defined by the Denavit-Hartenburg method. In the calibration process, the value of the joint variable for a given position

is considered fixed. The joint parameter or error is to be determined from its nominal value which of course is zero. With this in mind, the transformation T_o^1 would, in part, consist of a rotation $Rot(z_o,\delta\theta_1+\theta_1)$ which is equivalent to the product of $Rot(z_o,\delta\theta_1)$ and $Rot(z_o,\theta_1)$. The first rotation, $Rot(z_o,\delta\theta_1)$, is between two fixed frames. Denoting the intermediate frame as $F^{o'}$, then a RPYT transformation made up of six fixed parameters, $T_M^{o'}$, could be developed which would be equivalent to the product of T_M^o and $Rot(z_o,\delta\theta_1)$ which is made up of 7 fixed parameters. Therefore $\delta\theta_1$ is not independent and cannot be individually identified. As a final note, F^o is considered to be at the actual zero position, $F^{o'}$ in the preceding discussion, T_M^o is a transformation between F^M and F^o as now defined. Additionally, one reason for defining the parameter d_o to be zero is to eliminate the dependency that clearly would exist between the z translation in T_M^o and any subsequent translation d_o along the same z axis in T_o^1.

As described earlier, F^o is placed with the z_o axis coincident with joint axis 1 and directed upward in accordance with the right hand rule. Joint axis 1 and 2 are nominally coincident and perpendicular. F^1 is allocated in accordance with standard Denavit-Hartenburg with z_1 aligned with joint axis 2 and x_1 perpendicular to joint axis 1 and 2. With this allocation, parameters a_o and α_o are nominally zero and -90° respectively. Joint axis 2 and 3 are nominally parallel so F^2 is assigned using the modified Denavit-Hartenburg method. As

illustrated in Figure 16, this places F^2 external to the manipulator. Due to the orthogonal nature of the axis thus far, θ_2, α_2 and β_2 are all nominally zero with a_2 nominally 431.85 mm as shown. Joint axis 3 and 4 are nominally perpendicular and offset a perpendicular distance of nominally 20.33 mm which become a_3 in T_2^3. The z axis of F^3 is aligned with joint axis 4 with an offset distance, d_3, of 149.09mm. Note that as illustrated, link 3 is shown with θ_3 equal to approximately 90°. Therefore, a_3 is in the negative x direction and α_3 is nominally 90°. The z axis of F^4 is placed in the direction of joint 5 which is nominally coincident with and perpendicular to joint axis 4. Consequently, θ_4 and a_4 are nominally zero and α_4 is -90° with the chosen direction of z_4. The distance between F^3 and F^4 along joint axis 4, which corresponds to d_4, is 433.0 mm. F^5 is placed so that z_5 lies in the direction of joint axis 6 which is again nominally perpendicular to joint axis 5. Therefore, θ_5, d_5 and a_5 are nominally zero and α_5 is nominally 90° for the chosen direction of z_5. According to standard Denavit-Hartenburg methodology, a sixth frame would be placed with its origin coincident with the origin of F^5. This would then be followed by a six parameter transformation to the end effector frame. However, in a similar manner as before, the parameters of T_6^E would be dependant on all four parameters of T_5^6 and hence not independently identifiable. Therefore, F^5 will be considered the last link frame and T_5^E will be a RPYT transformation

containing a rotation about z_5 equivalent to the sum of the joint variable θ_6, joint parameter $\delta\theta_6$ and any other fixed rotation necessary for proper alignment at the end effector.

Table 1 summarizes the parameters for the five modified Denavit-Hartenburg transformations. A tabular presentation of this form is usually referred to as a kinematic parameter table. The bold elements comprise the 18 identifiable parameters of the manipulator. The other elements are zero as defined previously. The additional 12 parameters of the 30 parameter model are the variables of T_M^0 and T_5^E which are dependant on the location of the external reference frame and the geometry of a particular end effector. The parameters of Table 1 were utilized in each of the following experiments. The makeup of T_M^0 and T_5^E will be described for each specific case.

TABLE 1. PUMA 560 KINEMATIC PARAMETER TABLE

T	θ_i°	d_i (mm)	a_i (mm)	α_i°	β_i°
0→1	0.0	0.0	**0.0**	**-90.0**	**0.0**
1→2	**0.0**	0.0	**431.85**	**0.0**	**0.0**
2→3	**0.0**	**149.09**	**-20.33**	**90.0**	0.0
3→4	**0.0**	**433.00**	**0.0**	**-90.0**	0.0
4→5	**0.0**	**0.0**	**0.0**	**90.0**	0.0

B. NUMERICAL SOLUTIONS UTILIZING IMSL ROUTINE ZXSSQ

1. Introduction

The IMSL routine ZXSSQ is a Levenberg-Marquardt algorithm for the solution of non-linear least squares problems. The general problem statement follows

- Minimize: $f_1(\mathbf{x})^2 + f_2(\mathbf{x})^2 + \cdots + f_M(\mathbf{x})^2$
- Over: $\mathbf{x} = [\, x_1, x_2, \cdots, x_N \,]$

At the n^{th} iteration, an estimation of $\mathbf{x}^{n+1}$ is calculated using a numerical estimate of the Jacobian. The Jacobian estimate is calculated by a forward or central finite difference method.

The routine requires a user supplied function for calculation of the $f_i(\mathbf{x})$ functions. An initial estimate of $\mathbf{x}$ is supplied to the routine by the main or calling program along with convergence criteria. Three convergence criteria are available:

- NSIG: The first convergence criteria is satisfied if on two successive iterations, the parameters agree to NSIG significant digits.
- EPS: The second convergence criteria is satisfied if the residual sum of squares for two successive iterations is less than EPS.
- DELTA: The third convergence criteria is satisfied if the euclidean norm of the estimated gradient is less than DELTA.

Satisfaction of any of the three criteria will halt program execution and a number of parameters are returned to the calling program including the final estimate of $\mathbf{x}$, the final value of each $f_i(\mathbf{x})$, the residual sum of squares in variable SSQ and the satisfied convergence criteria. Three variations

of the algorithm are selectable by defining option parameter IOPT. If the residual sum of squares is close to zero then setting IOPT to zero (Brown's algorithm without strict descent) will usually perform satisfactorily. This setting of IOPT was used for all applications of the routine due to the problem formulation.

The general program flow is illustrated in Figure 17. An initial estimate, $\mathbf{x}^0$, is supplied from the calling program along with several parameters including the convergence criteria and algorithm option. The routine then calls the user supplied subroutine N times where N is the number of elements of $\mathbf{x}$ in order to calculate the finite difference gradient approximations. ZXSSQ then calculates a new estimate, x^{n+1}, and then calls the user supplied routine to calculate $f_i(x^{n+1})$. The process repeats until any one of three convergence criteria is satisfied.

Although mathematically equivalent, two different formulations of the problem statement for implementation of ZXSSQ are used in this thesis. Each formulation is described in the following two sections.

2. Data Fitting

In the calibration process, measurement of the end effector pose, full or partial, are made and then this data is used to adjust or fine tune the kinematic model parameters. A simplistic view of this problem consists of holding the usual model variables fixed and varying the constants. A simple

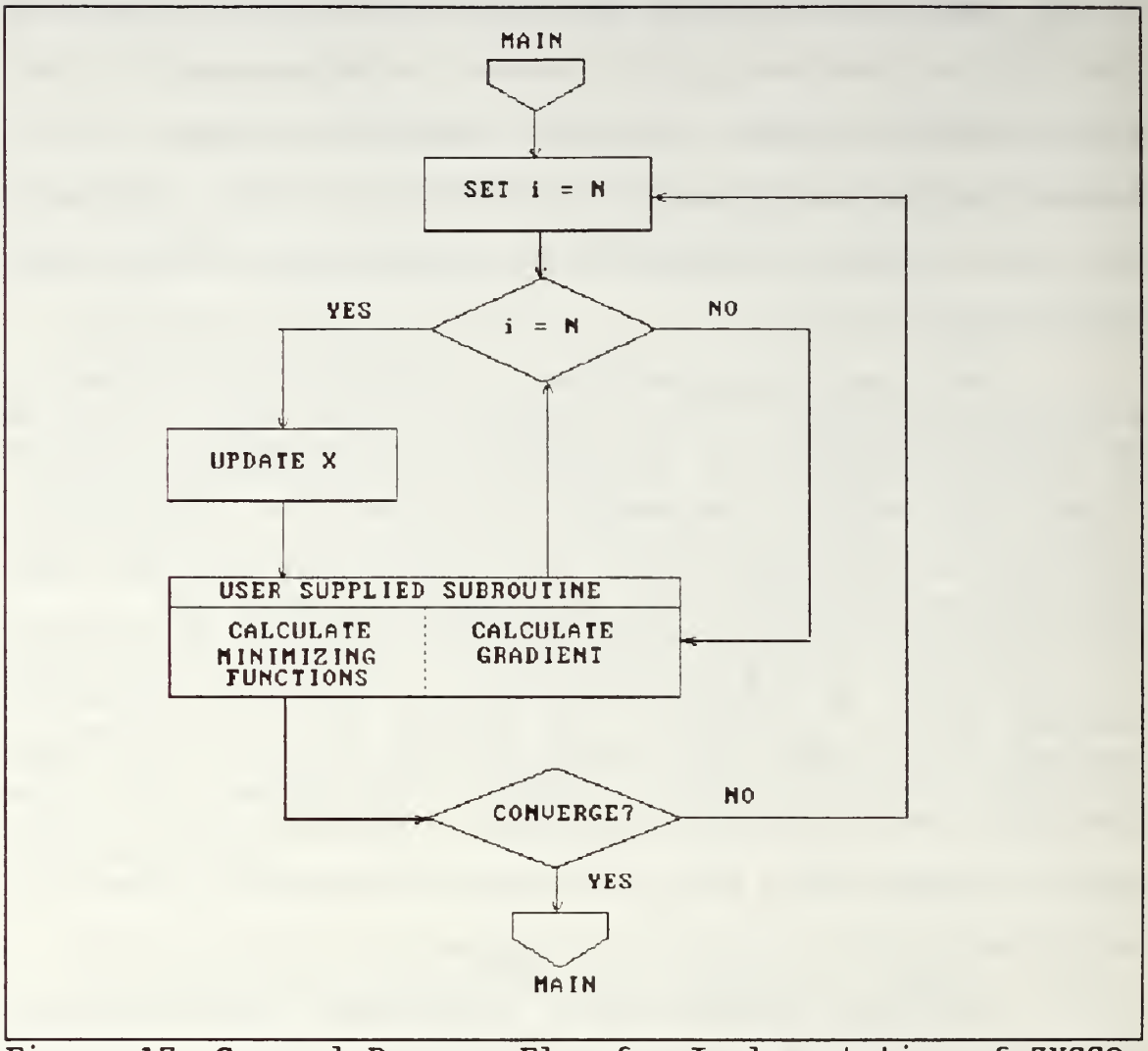

Figure 17. General Program Flow for Implementation of ZXSSQ

example follows.

Suppose the mathematical model for a physical process is assumed to have the form of Equation 32.

$$g(x,y) = a_1x^2 + a_2x + a_3y^2 + a_4y + a_5 \qquad (32)$$

The values of the constants of Equation 32 are arrived at based on theory, physical laws, reasonable assumptions, or design parameters and assumed to be greater than zero for this example. It is desired to know the actual values for these

constants. For application of ZXSSQ, the problem can be formulated in the following manner.

Measure g(x,y) for m known values (x,y). Denote these measured g(x,y) as h_1, h_2, • • • , h_m where h_i corresponds to (x_i,y_i). Compute $g_i(x_i,y_i)$ from Equation 32 and then let $f_i = g_i - h_i$. In this problem, the five a_i are the variables. The problem statement is then

- Minimize: $[f_i(\mathbf{x})]^2$
- Over: $\mathbf{x} = [\ a_1,\ a_2,\ a_3,\ a_4,\ a_5\ ]$

The problem is now in a form suitable for application of ZXSSQ. The calling program would provide the assumed values of the constants a_1, a_2, a_3, a_4 and a_5 as initial value $\mathbf{x}^0$. The user supplied function would compute values of g_i based on (x_i,y_i), which are fixed for a given i, and $\mathbf{x}^{n+1}$, the updated or perturbed values of a_1, a_2, a_3, a_4 and a_5. Updated values of f_i are then calculated using the new values of g_i. Note that this problem consists of 5 unknowns, the coefficients of g in Equation 32, and each measurement provides one known value h_i. Assuming that the actual system does satisfy a paraboloid relationship and in the absence of measurement noise, then at least 5 measurements are necessary for a solution.

3. Optimization

ZXSSQ can be employed in an optimization scheme in which one or more analytic expressions are minimized. In this case, the constants are fixed and the variables are perturbed. A simple example follows.

Clearly, a simple analytic solution can be obtained for the minimum of the paraboloid of the previous example. However, for illustrative purposes, consider g to be described by two functions g_1 and g_2 as shown in Equations 33.

$$\begin{aligned} & g(x,y) = g_1(x,y) + g_2(x,y), \\ \textit{where} \quad & g_1(x,y) = a_1x^2 + a_3y^2 + a_5 \\ & g_2(x,y) = a_2x + a_4y \end{aligned} \tag{33}$$

In this example, the problem can be stated,

- Minimize: $(g_1(\mathbf{x}))^2 + (g_2(\mathbf{x}))^2$
- Over: $\mathbf{x} = [\ x,\ y\ ]$

which is now in the proper form for application of ZXSSQ. In this case, the calling program must supply an initial guess of the vector $\mathbf{x}^\circ$. If the minimized functions are unimodal, any reasonable value of $\mathbf{x}^\circ$ should allow convergence. Problems associated with non-unique solutions can be addressed in a number of different ways and are somewhat problem dependant. A good initial estimate of $\mathbf{x}^\circ$ when known may suffice.

Another problem which may arise in an optimization problem is that of proportionality. It is clear that it would suffice to minimize g(x) rather than the two functions g_1 and g_2 although both approaches should have similar results. The particular problem formulation chosen in this case demonstrates two ways in which a problem can be disproportional. First of all, for values of x and y much greater than or less than one, small changes in x and y may have much greater effect on g_1 than on g_2. Secondly, if a_5 is much greater than zero, then at or near the minimum, g_1 may be

much greater than g_2. Disproportionate problem formulation can lead to numerical instability or inaccuracies. Scaling techniques, such as dividing g_1 by a_5, or reformulating the problem statement can reduce or eliminate difficulties associated with proportionality.

C. KINEMATIC MODEL PARAMETER IDENTIFICATION METHODOLOGY

1. General Scheme

Without loss of generality, the general scheme will be described considering the previously described 30 parameter PUMA kinematic model and a measurement system capable of full pose measurement. Less capable measurement systems generally result in a reduction in the number of identifiable parameters in the model. However, the general scheme remains the same and the specific differences will be addressed on a case by case basis.

Given the 30 parameter kinematic model based on nominal values, actual parameter identification or calibration is performed in the following manner. Measurement of the end effector pose is made and the joint variable values and the measurement pose data are both recorded. The manipulator joints are varied, additional measurements made and recorded, and the process repeated until a sufficient data base is collected. Sufficient has at least two meanings in this case. First of all, note that the model must reflect the capabilities of the measurement system. If the measurement system is capable of measuring full pose as in this case, each

measurement consists of six knowns, three positions and three orientations. For an N parameter model, a minimum of N/6 measurements are required. However, some measurement noise is inevitable and some larger number of measurements must be taken to achieve a desired accuracy. Some additional factors, some of which will be discussed later and some which are issues for further research, must be considered when attempting to quantify the meaning of a sufficient data base.

The pose data can be recorded in matrix form and will be denoted T^{EAi} where E refers to end effector as before, A refers to actual or measured and i used to denote a specific measurement. The forward kinematic solution can be computed based on the nominal parameters and the i^{th} set of joint angles and stored in T^{ECi} where C refers to the calculated value. Recall that when calculating the forward solution, both the joint variable and joint parameter must be taken into account. A matrix ΔT_i can be computed from the difference of T^{EAi} and T^{ECi}. As described by Paul [Ref. 9], a differential transformation matrix has the following form

$$\Delta = \begin{bmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & -\delta_x & dy \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad (34)$$

The Δ matrix of Equation 34 is a good approximation for small joint variable variations and provides a reasonable approximation if all nominal parameters are "close" to their

associated actual values. ΔT_i will not necessarily have the odd symmetry of the upper 3 by 3 submatrix of the Δ matrix of Equation 34. However, an average value of the magnitude of δ_x, δ_y and δ_z can be computed as shown in Equation 35

$$\delta_x = \left| \frac{t_{32} - t_{23}}{2} \right|$$
$$\delta_y = \left| \frac{t_{13} - t_{31}}{2} \right| \qquad (35)$$
$$\delta_z = \left| \frac{t_{12} - t_{21}}{2} \right|$$

where t_{ij} is the i^{th} row and j^{th} column entry of ΔT_i.

This problem can now be stated in a form acceptable for implementation of ZXSSQ:

$$\begin{aligned} &\textit{Minimize:} \quad \sum_{i=1}^{n} \sum_{j=1}^{6} (f_{ij}(\boldsymbol{x}))^2 \\ &\textit{Over:} \quad \boldsymbol{x} = [\phi_m, \theta_m, \psi_m, x_m, y_m, z_m, a_1, \alpha_1, \theta_2, \cdots, x_E, y_E, z_E] \end{aligned} \qquad (36)$$

where x is vector of length 30 containing the kinematic parameters, N is the number of measurements, i corresponds to each of the N measurements and

$$\begin{aligned} f_{i,1}(\boldsymbol{x}) &= \delta_x \\ f_{i,2}(\boldsymbol{x}) &= \delta_y \\ f_{i,3}(\boldsymbol{x}) &= \delta_z \\ f_{i,4}(\boldsymbol{x}) &= d_x \\ f_{i,5}(\boldsymbol{x}) &= d_y \\ f_{i,6}(\boldsymbol{x}) &= d_z \end{aligned} \qquad (37)$$

Note that δ_x, δ_y, δ_z are the average values of ΔT_i as computed in Equations 35 and d_x, d_y and d_z are the t_{14}, t_{24} and t_{34} elements of ΔT_i.

2. Program ID6 (Generic Version)

The program ID6 was used to compute the numerical solution of the calibration problem. A flowchart for the program is shown in Figure 18. The program is essentially the same for all three experiments performed in this thesis. Specific differences will be addressed as each experiment is presented.

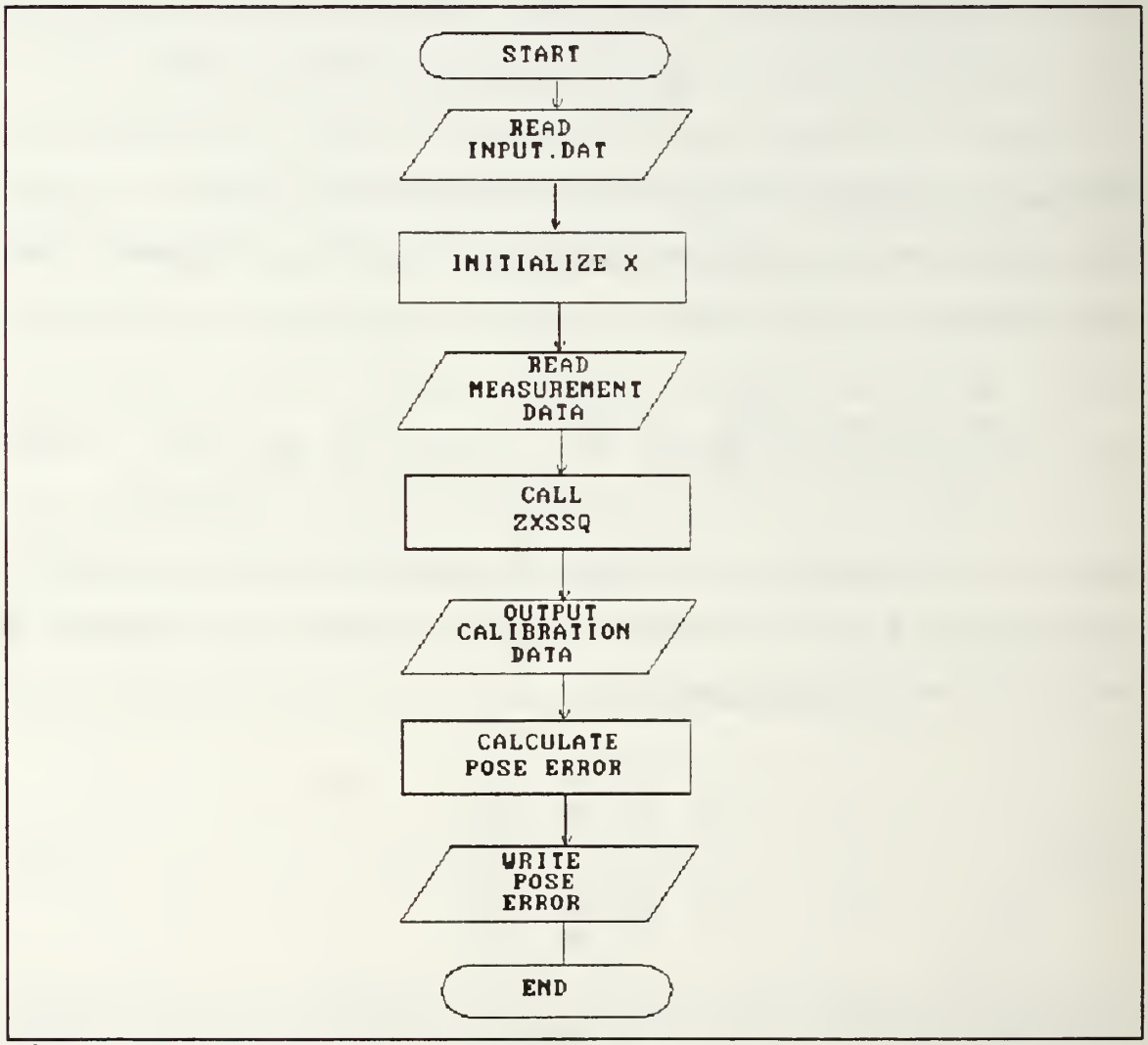

Figure 18. Program ID6 Flowchart

The program reads an input file consisting of the kinematic parameters for the particular model. These parameters are used to initialize the vector **x** for the subroutine ZXSSQ. The measurement data set is read from a file and assigned to several different vectors. One vector contains the full or partial pose data of the measurements and is of length N where N is the number of measurements. In the case of full pose measurement, this vector has dimension 4 by 4 by N and contains N T^E matrices. The joint variable data is stored in six 1 by N vectors, one vector for each joint variable. The various parameters and options associated with ZXSSQ are initialized and then the subroutine is called.

Upon ZXSSQ termination, the updated values of the kinematic parameters are output to a file along with the residual values of position and orientation as appropriate.

Associated with each version of ID6 is the "user defined subroutine" called by ZXSSQ titled PUMA_ARM. The subroutine is passed the current value of **x** for either gradient estimation or computing the updated value of the minimizing functions. The joint variable data and measurement data are passed via a common block. The parameters are updated by this current value of **x**. N iterations of the following calculations are performed where N is the number of measurements.

During the i^{th} iteration, the joint variables for measurement i are added to their corresponding joint

parameters. The forward kinematic solution is calculated based on the current value of the parameters. The minimizing functions for measurement i are calculated based on the forward solution and the measurement data as described in the preceeding section. The process is repeated until i = N at which time the function values are passed back to ZXSSQ.

III. THREE CALIBRATION TECHNIQUES

A. COORDINATE MEASURING MACHINE FULL POSE CALIBRATION

1. Physical Description of the Measurement System

a. The Coordinate Measuring Machine

The Coordinate Measuring Machine, CMM, is illustrated in Figure 19. The horizontal base assembly consists of a fixed base and a carriage which is free to move along the length of the assembly. This direction is usually defined as the x axis. The carriage is held in alignment by two guide bars which have precision racks machined on their surfaces. The racks provide motion through a rack and pinion arrangement and rotation of the x axis knob as shown in Figure 19. Optical encoders in the carriage assembly provide displacement measurements. The vertical column is constructed and functions in a manner similar to the x axis carriage. This direction is designated as the y axis. Motion in the z direction is accommodated by the horizontal assembly mounted on the y axis carriage as shown, and is constructed and functions in the same manner as the other two axes.

A display unit, not shown, is provided and is capable of indicating either in inches or millimeters. The CMM is capable of 0.01 mm accuracy on all axis. The display output can be zeroed for all axes simultaneously by depressing the "ALL ZERO" pushbutton or each axis can be zeroed separately by

pressing the appropriate axis "zero" pushbutton and contact with a touch probe.

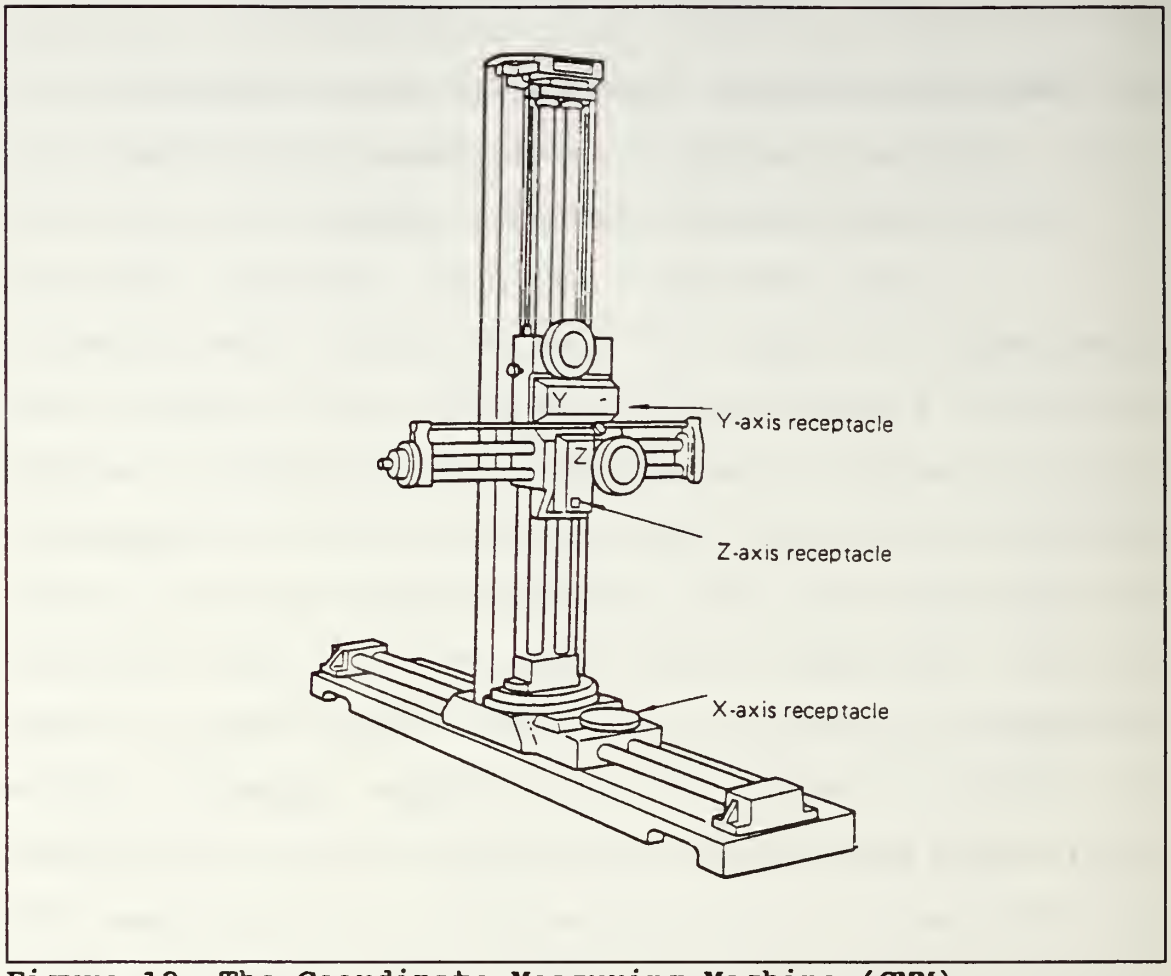

Figure 19. The Coordinate Measuring Machine (CMM)

The touch probe, illustrated in Figure 20, is mounted on the end of the z axis of the CMM. The touch probe tip is a machined sphere of 3.0 mm diameter. When the probe comes in contact with an object, the indicator will illuminate and the display unit readout will hold its present reading until the probe is no longer in contact. When the display unit "zero" is set for a particular axis, the axis readout is zeroed by touch probe contact. This is a useful feature

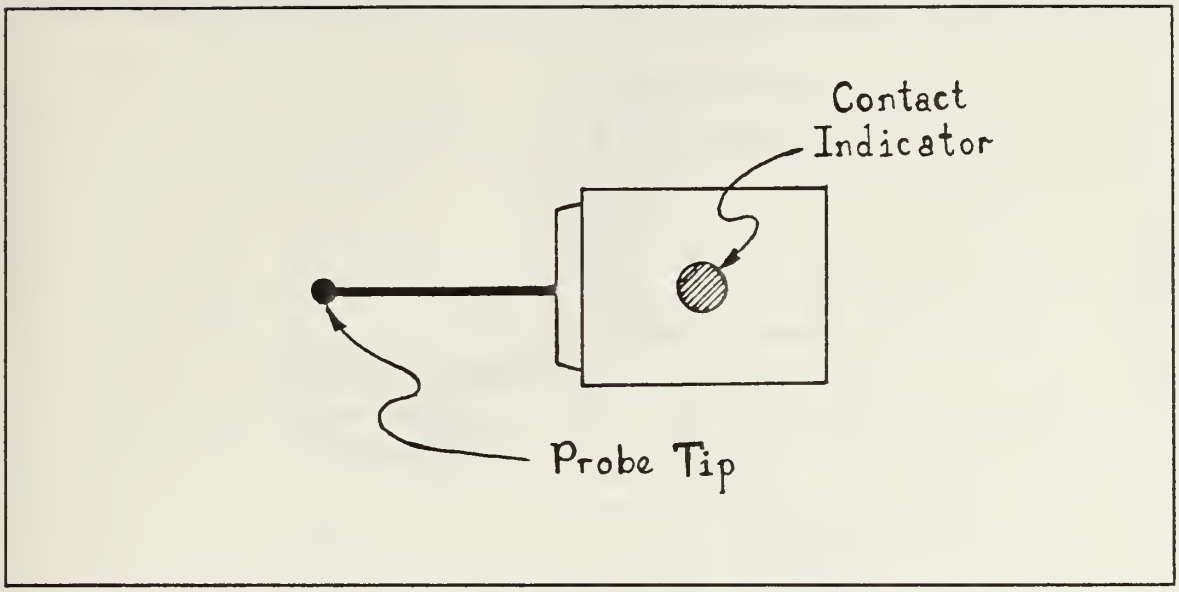

Figure 20. CMM Touch Probe

because it provides a method of establishing a reference frame external to both the manipulator and the CMM, which will be described in the following sections.

A machined cube, similar to that illustrated in Figure 21, was mounted in the common working volume of the PUMA and the CMM. The faces of the cube were nominally aligned with the axis of the CMM. A corner of the cube was chosen as the reference point of the measurement system which eliminated the need for absolute alignment of the cube faces with parallel planes formed by the CMM x, y and z axis. Using the touch probe zero reference feature, a reference point could be established. For example, with the display unit 'x-zero' enabled, the probe can be placed near the y-z face of the cube in close proximity to the reference corner and then slowly moved in the x direction until contact is made and the x=0 reference established.

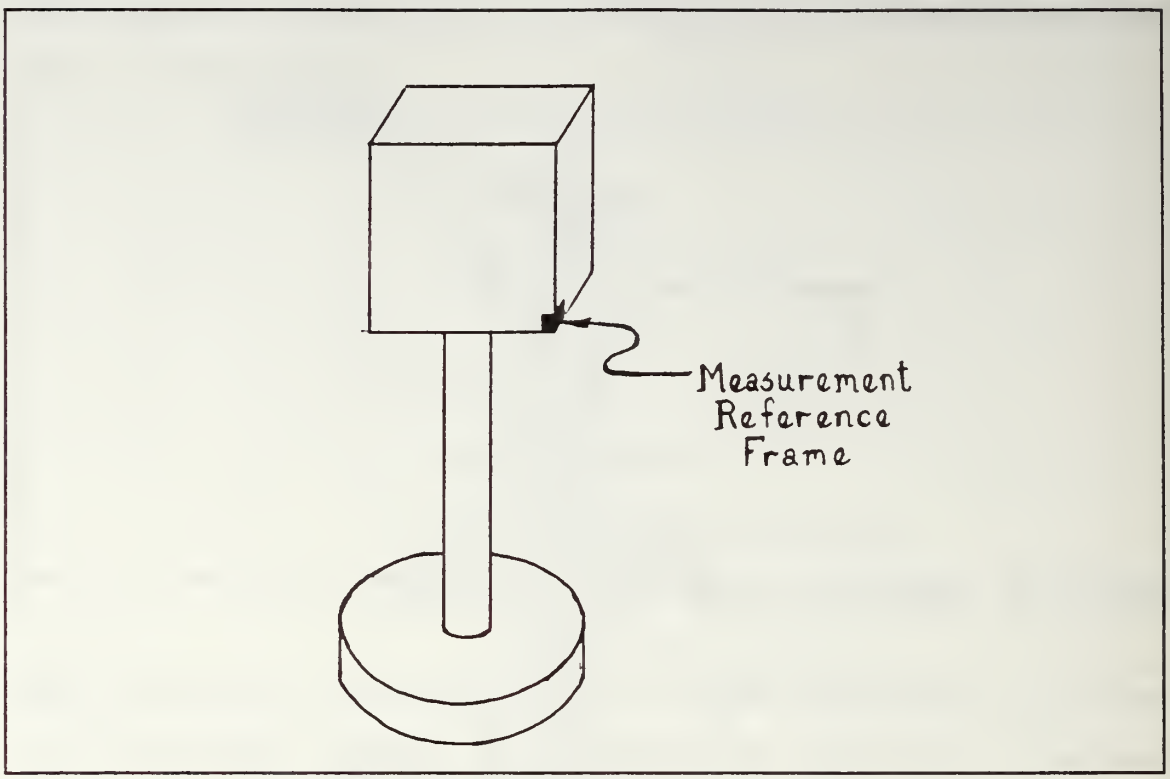

Figure 21. Measurement System Reference Cube

b. Manipulator End Effector

To calibrate a 30 parameter model of the PUMA 560 requires measurement of the pose of the end effector. Since the CMM is only capable of position measurement, the end effector must be of some known geometry such that orientation can be calculated from a series of position measurements. The end effector illustrated in Figure 22 was used in this experiment.

The five machined tooling balls of radius 6.35 mm are mounted orthogonally to the circular plate and post as shown. The fabrication process guaranteed orthogonality of the fixture, but the specific location of each tooling ball on its respective axis was not guaranteed. These positions were

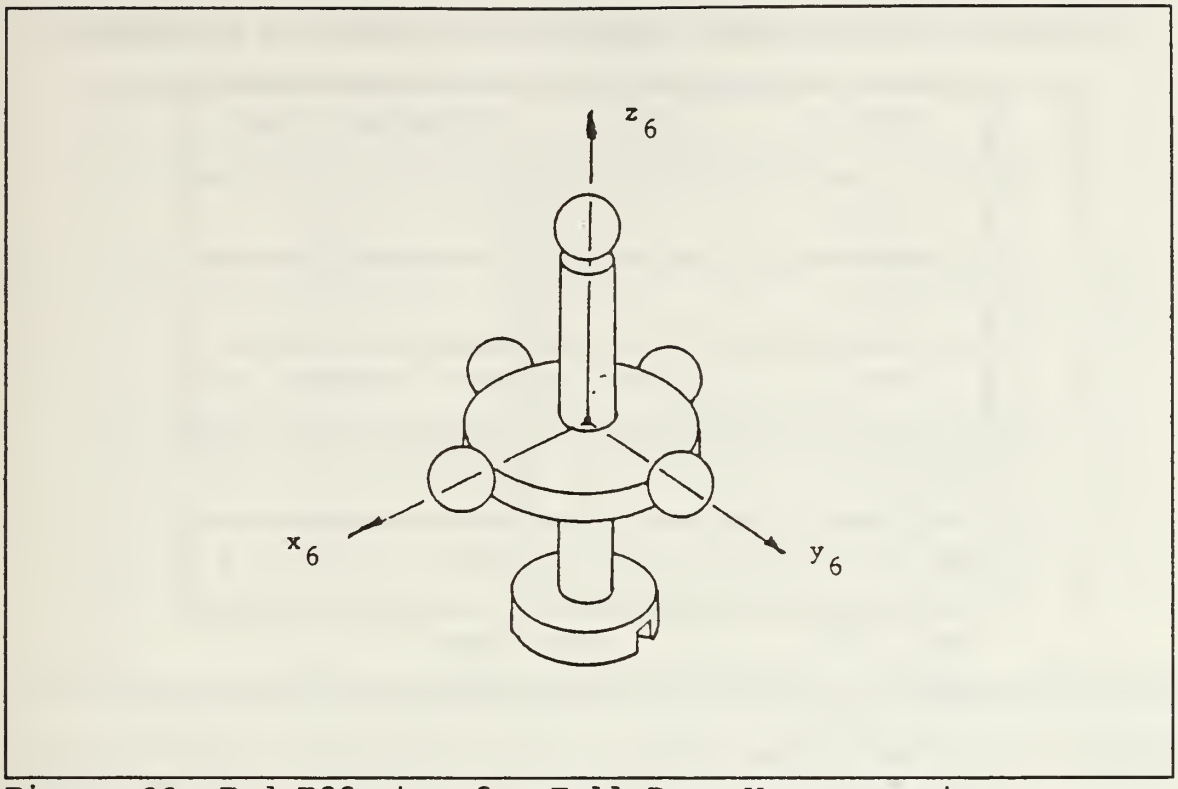

Figure 22. End Effector for Full Pose Measurement

determined by a statis calibration with the CMM. The balls were numbered one through five and their corresponding axis and distance along the axis to the origin of the coordinate frame are listed in Table 2. The lower flange is mated with the PUMA end effector mounting flange.

2. Theory

a. Closed Chain Kinematic Model

The link parameter table listed previously in Table 1, is used for manipulator transformations in all three experiments. The only difference in the models for each experiment is in the identifiable parameters of T_H^0 and T_5^E.

TABLE 2. END EFFECTOR BALL DESIGNATIONS AND DIMENSIONS

Ball #	Axis	Distance (mm)
1	z	51.111
2	x	50.740
3	y	50.740
4	-x	50.913
5	-y	50.988

Since the CMM is capable of fully defining its own coordinate system, and with the use of the previously described end effector, full pose measurement is possible and all 30 parameters of the model can be identified. Table 3 lists the nominal values for the 12 parameters of T_M^0 and T_5^E and are based on the nominal position of the measurement system reference point, orientation of the CMM with respect to the PUMA base frame and the nominal orientation of the end effector. The values are typed in bold to emphasize that they are all identifiable.

b. Developing Full Pose Data

To determine the pose of the coordinate frame defined on the end effector first requires knowledge of the coordinates of the center of the tooling balls with respect to F^M. Recall the relationship given in Equation 1 between the radius r of a sphere, its center at x_o, y_o, and z_o, and the

TABLE 3. KINEMATIC PARAMETER TABLE FOR T_M^O AND T_5^E

	T_M^O	T_5^E
ϕ	180.0°	90.0°
θ	0.0°	0.0°
ψ	90.0°	0.0°
●	-383.0 mm	0.0 mm
●	-383.0 mm	0.0 mm
z	474.0 mm	134.0 mm

coordinates of a point P on the surface of the sphere.

$$r = \sqrt{(x_0-p_x)^2 + (y_0-p_y)^2 + (z_0-p_z)^2} \qquad (38)$$

Since measurement of points on the surface of the sphere are possible with the CMM and the radius of the precision tooling balls is known, then three unknowns remain in Equation 1. This implies that a minimum of three measurements are required for a fully defined problem involving three non-linear equations with three unknowns. ZXSSQ can be employed for a solution with the following problem statement:

- Minimize: $\Sigma\ (f_i(\mathbf{x}))^2$
- Over: $\mathbf{x} = [\ x_o,\ y_o,\ z_o\ ]$

where

- x_o, y_o and z_o are the coordinates of the center of the sphere

- $f_i(x) = | r - r_i |$
- r is the radius of the sphere
- r_i is calculated from Equation 1 for the i^{th} measurement of a point on the tooling ball

Simulation and experiment showed that four measurements provided sufficient accuracy.

With the coordinates of the center of the tooling balls now available, determination of the pose can be developed. Recall that the coordinates of a point P described with respect to F^E can be transformed into F^M coordinates by

$$P_M = T_M^E \; P_E \tag{39}$$

Unfortunately, in this case, P_M and P_E are known and T_M^E is the unknown and the vector of the coordinates of P_E can not be inverted in order to solve for the transformation. However, as described in Paul [Ref. 10], all the coordinates of an object can be transformed from one frame to another simultaneously by composing a matrix whose columns are the coordinates of the object to be transformed and then pre-multiplying by the transformation matrix describing the frame. Note that the points are described by the usual augmented 4x1 vectors. Let P_M and P_E denote two matrices composed of the coordinates of the center of four balls, then

$$\begin{aligned} P_M &= T_M^E \; P_E \\ P_M \; P_E^{-1} &= T_M^E \end{aligned} \tag{40}$$

It may not always be possible to measure four balls for any given pose. However, with the given geometry, a

fourth location, orthogonal to three measured balls, can be synthesized by

$$P_4 = (P_2 - P_1) \times (P_3 - P_1) \qquad (41)$$

where the subscripts are not intended to imply any order or particular ball, but only three different balls. This calculation must be performed twice, once for points described with respect to F^M and once for points described with respect to F^o.

3. Simulation

a. Introduction

The calibration process is well suited to computer simulation for the following reasons:

- Experimental data simulation, including noise injection, is usually a straight forward process.
- The heart of the process is a numerical solution performed by computer.
- Analysis of the results is easily performed on computer.

Several advantages are offered by first performing a computer simulation:

- The identification algorithm can be tested.
- Trends in the accuracy of the solution when compared with the number of measurements based on predicted noise level can be identified.
- To some extent, the model can be validated during the simulation. For example, if dependant parameters are included in the model, the identification algorithm will not converge to the correct solution since no unique solution exists.

The general simulation scheme used for this experiment is illustrated in Figure 23. The programs and their associated input and output files are described in the following sections.

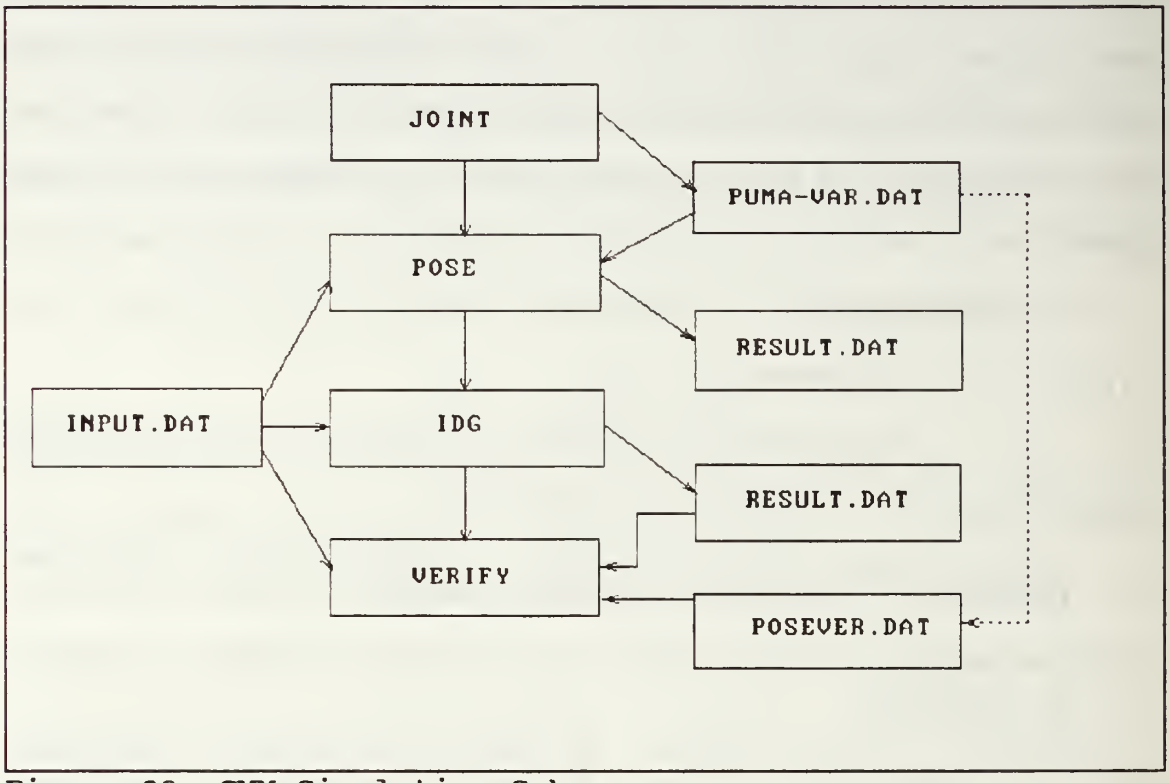

Figure 23. CMM Simulation Scheme

b. The Program JOINT

The program joint uses a Monte-Carlo method random number generator to produce random sets of six joint variable values. Three options for the range of joint motion for each joint are available. Full range, one-half of normal range, and one-quarter of normal range of joint motion can be selected and this allows simulation of the effects of a limited working volume. The number of sets of joint angles is interactively supplied to the program. The program writes the joint angle

sets to a file titled PUMA_VAR.DAT. The program is executed twice with the second output file renamed POSVER.DAT for use in a verification program which will be described later.

c. The Program POSE

Program POSE reads a file, INPUT.DAT, containing the following information:

- the previously described kinematic parameter table;
- the number of observations or measurements simulated;
- the number of model parameters;
- length and angular offsets;
- length and angular noise scaling constants.

The length and angular offsets are added to each of the identifiable parameters so that a known model, different from the nominal model, can be used to generate the simulated pose measurements. POSE reads the random sets of joint angles from the file PUMA_VAR.DAT. The joint variable angles are added to the offset joint parameters and a forward kinematic solution is calculated. The resulting solution, a 4x4 T^E matrix based on offset parameters, is stored in a file titled PUMA_POS.DAT along with the corresponding set of joint variable data. Prior to storage, noise can be injected into the joint variable angles, and separately into the orientation and position elements of T^E. The random noise is calculated by scaling the output from a Monte-Carlo random number generator and then added to the desired parameters.

d. The Program CID6

The previously described file INPUT.DAT is read and stored in CID6. The identifiable nominal parameters are used to initialize the ZXSSQ **x** vector. The simulated pose measurements and joint variable data are read from PUMA_POS.DAT and stored in their respective arrays. ZXSSQ parameters are initialized and then the subroutine is called.

ZXSSQ and the external subroutine PUMA_ARM perform the identification process as described in the Kinematic Model Identification Methodology Section general scheme, and summarized in Equation sets 36 and 37.

After termination of ZXSSQ, the calibrated link parameter table is written to a file titled RESULT.DAT. Up to this point, the simulation version of this program is identical to the experimental version. However, in the simulation, the actual model parameters are known since they were formed by adding a known offset to the nominal parameters. Therefore, the accuracy of the calibrated parameters can be determined by again adding the offset values to the nominal parameters and comparing these values to the calibrated values. An rms value for both length and angular parameters is computed and also written to RESULT.DAT along with the residual from ZXSSQ.

e. The Program VERIFY

As a final test of the accuracy of this method, the program VERIFY computes a forward solution based on the

actual model parameters and the calibrated parameters, and compares the resulting poses. To compute the actual pose, VERIFY reads file INPUT.DAT and, as before, adds the length and angular offsets to the appropriate kinematic parameters. The calibrated kinematic parameter table is read from the file RESULT.DAT. Two separate forward solutions are calculated based on the two sets of kinematic parameters and sets of random joint angles produced by JOINT and read from the file POSVER.DAT described earlier. An average orientation and position error is calculated from the difference between the two T^E matrices resulting from the forward solution calculations.

4. Experiment

a. Data Acquisition

As noted in a preceding section on developing full pose data, four measurements with the CMM are required to locate the center of one tooling ball of the end effector. Additionally, the centers of three tooling balls were required to develop a full pose measurement T^E. A program, CMMPOSE, was written to convert the CMM position measurements into pose measurements.

CMMPOSE is given three sets of four CMM position measurements interactively along with the associated ball numbers. A subroutine, BALL, calculates the center of each tooling ball using the previously described algorithm. After the center of each ball is calculated, the residual is

displayed on the terminal and the user is prompted to either keep or reject the value based on the residual. Residuals greater than 10^{-6} were rejected and a second set of measurements taken. This process allows the user to reject poor data, probably caused by incorrect reading of the CMM display or data entry errors. CMMPOSE then synthesizes a fourth spatial position, composes the two position matrices described earlier and then computes the measured pose. An orthogonality check is performed by

$$\boldsymbol{n} \cdot \boldsymbol{o}, \quad \boldsymbol{n} \cdot \boldsymbol{a}, \quad \boldsymbol{o} \cdot \boldsymbol{a} \tag{42}$$

where n, o and a are vectors corresponding to the first three columns of T^E and "·" is the dot product operator as before. Since these vectors correspond to rotated coordinate axes then their dot products should be zero. If the orthogonality check passes, then the program stores the T^E matrix and the current joint angle variables which are interactively input by the user in a file PUMA_POS.DAT. Joint angle variables of the PUMA 560 can be obtained from the PUMA 560 console by typing "where" on the keyboard.

Data was acquired by first zeroing the CMM at the reference point as described earlier. The CMM was slowly positioned so that light contact was made with the surface of one of the tooling balls. Care was exercised to insure that the direction of approach was nearly normal to the tangent planes of the tooling ball and to the touch probe tip, at their contact point. This reduces possible errors caused by

deflection of the tip away from the point of contact. This process is repeated with the position data supplied to CMMPOSE, along with the corresponding joint variable angles. Two operators, one operating the CMM and positioning the PUMA with the teach pendant,[1] and another entering data at a console can expect to make one full pose measurement in approximately 10 minutes. Figure 24 illustrates the CMM ready for a position measurement to be taken.

b. Parameter Identification and Verification

Program CID6 was modified as described earlier for experimental data. A total of 44 poses were collected and the entire set of poses were used to calibrate the PUMA. Table 4 lists the nominal and calibrated values of the kinematic parameters. Length and angular parameters are reports in units of millimeters and degrees, respectively.

Since the actual parameters are not known as they are in the simulation, a program such as VERIFY cannot be used. However, to better evaluate the accuracy of the resulting calibration, the set of 44 poses and associated joint variable angles were divided equally into two sets, S_1 and S_2. S_1 was used to perform a calibration. S_2 was then used in a verification process in the following manner. Forward

---

[1]The PUMA 560 can be controlled by computer or manually controlled by the teach pendant. The teach pendant allows an operator individual positioning of each joint when operated in "joint mode". Additional teach pendant modes are available for positioning the end effector with respect to the manipulator tool frame or base frame.

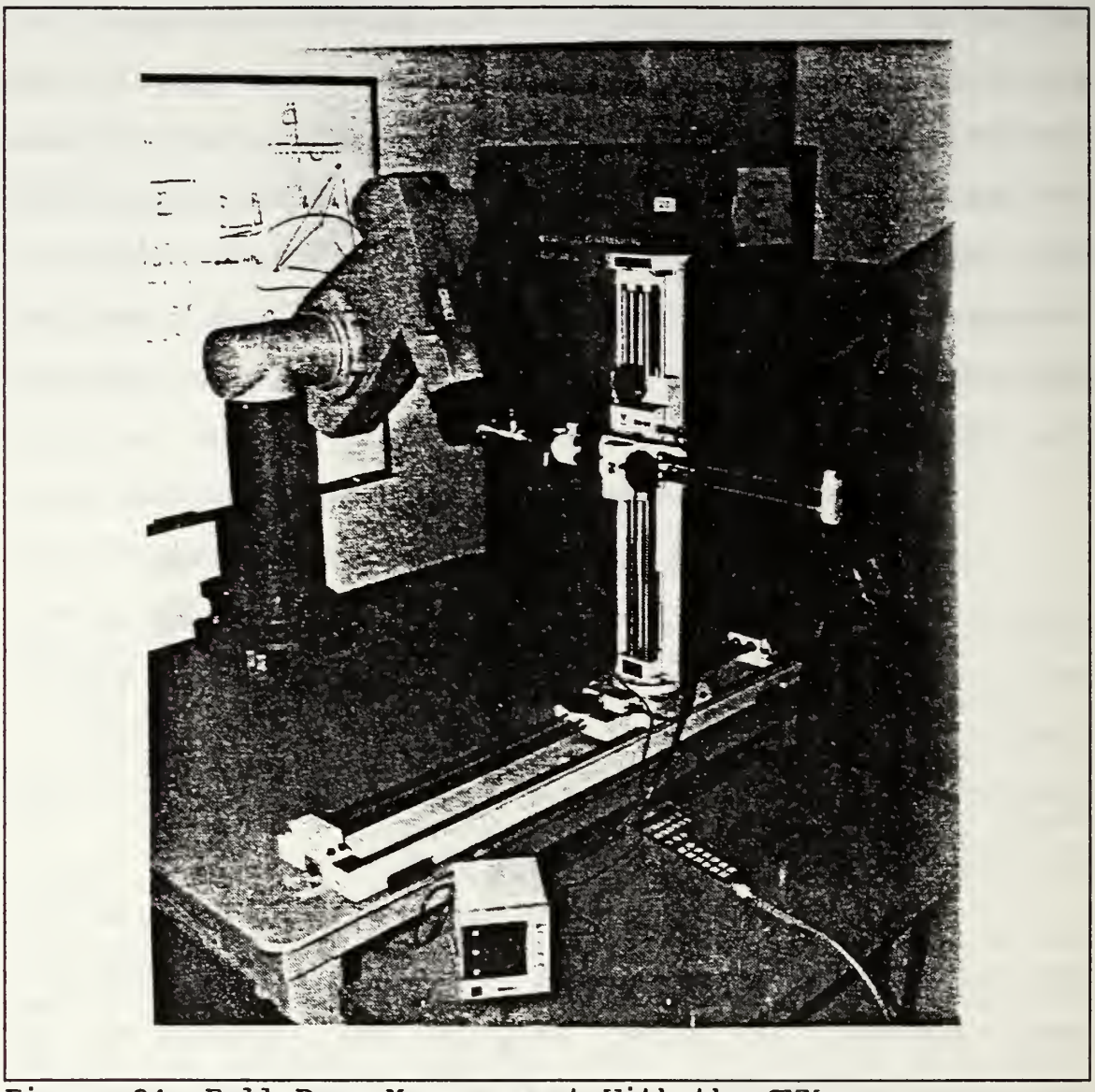

Figure 24. Full Pose Measurement With the CMM

solutions using the joint variable angles of S_2 and the parameters calibrated with S_1 were calculated. The difference between these calculated poses and the poses of S_2 provides a good indication of the improvement in accuracy. An rms value of 0.3 mm was calculated for the position error.

TABLE 4. NOMINAL AND CALIBRATED KINEMATIC PARAMETERS

Parameter	Nominal	Calibrated
ϕ_M	180.0	179.9579
θ_M	0.0	1.5120
ϕ_M	90.0	89.0219
x_M	-394.0	-393.9838
y_M	-383.0	-405.0608
z_M	474.0	466.8381
a_1	0.0	-0.04923
α_1	-90.0	-89.9977
$\delta\theta_2$	0.0	-0.4888
d_4	431.9	432.1216
α_2	0.0	-0.0303
β_2	0.0	-0.01515
$\delta\theta_3$	0.0	-1.2069
d_3	149.1	149.1455
a_4	-20.3	-19.2270
α_3	90.0	90.0512
$\delta\theta_4$	0.0	-0.9144
d_4	433.0	432.8899
a_4	0.0	0.0040
α_4	-90.0	-89.9909
$\delta\theta_5$	0.0	2.2364
α_5	0.0	-0.6629
a_5	0.0	-0.0258
α_5	90.0	89.9345
ϕ_E	90.0	91.2400
θ_E	0.0	-0.0979
ψ_E	0.0	-0.0575
x_E	0.0	0.1863
y_E	0.0	-0.2329
z_E	134.0	133.1557

B. THE MODIFIED LINEAR SLIDE METHOD

1. Introduction

In an experiment performed by Potter [Ref. 11], a PUMA 560 was successfully calibrated using a measurement system referred to as a linear slide. The original experimental

measurement system and the modification will be described in the following sections. Although the method offered several apparent advantages over the CMM calibration method, the resulting accuracy was less by a factor of three. Two possible explanations for the resulting loss of accuracy are increased measurement noise due to loading effects and limited range of joint rotation. The modification described in the following was attempted to improve the range of joint motion and thereby increase the overall calibration accuracy.

2. Physical Description of the Measurement System

In the original experiment, the vertical post (y and z axis) of the CMM were removed. A plate was manufactured which mates flush with the end effector flange of the PUMA and flush with the X carriage of the CMM in the position vacated by the vertical post. In this configuration, the orientation of F^E is fixed and the end effector restricted to linear motion along the x axis of the CMM base. The experimental setup is illustrated in Figure 25. As shown in the figure, the CMM base was placed on a ramp. The ramp was necessary to ensure adequate rotation of all PUMA joints. For example, due to the design of the PUMA, if the slide was placed horizontally on the table, essentially no rotation of the wrist, joint 4, occurs during translation along the axis of the CMM base. With no wrist rotation, the PUMA becomes a five degree of freedom manipulator and the kinematic model must be changed to reflect this apparent loss of a joint. Additional

problems may be encountered with respect to identification when joint rotation is limited and these problems will be discussed in greater detail later. The position and orientation of the ramp, as shown, offered reasonable joint excursion for all joints as the end effector moved along the slide.

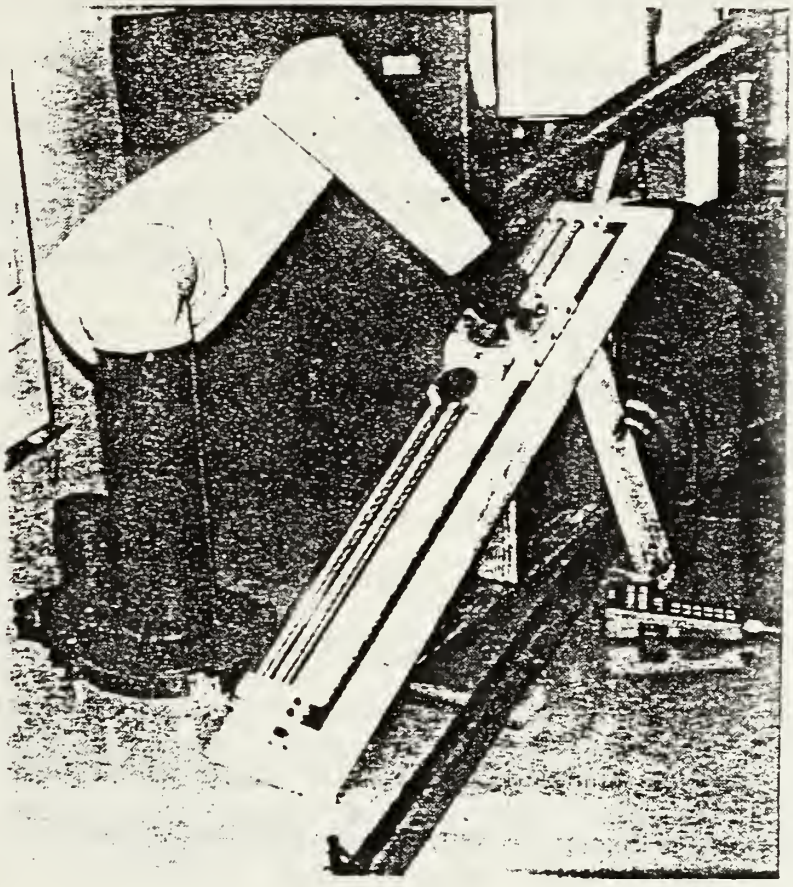

Figure 25. PUMA Calibration with a Linear Slide

To improve the range of joint rotation, a ball joint was placed between the PUMA end effector flange and the X carriage of the CMM. A drawing of the fixture can be seen in Figure 26. In this configuration, the complete range of motion of joints four, five and six could be achieved. Approximately 20 additional degrees of rotation could be achieved for joints two and three. However, joint one rotation remained essentially unchanged.

Note that for both methods, the end effector of the PUMA is physically constrained, which requires the PUMA to be placed in "free" mode.[2] Consequently, the PUMA must be supported by an operator during calibration to prevent it from collapsing and damaging itself or the measurement system.

3. Closed Chain Kinematic Model

The closed chain kinematic model for the modified linear slide method varies significantly from the original linear slide method, which will be described in another section of this thesis. As noted previously, the kinematic parameters listed in Table 1 are used to describe the manipulator. It is the parameters of T_M^0 and T_5^E that must be determined. First, consider the ball joint attached to the end effector flange. A ball joint has three degrees of freedom,

---

[2]Each joint of the PUMA is equipped with a coarse and fine joint encoder and servomotor. Additionally, joints one, two and three are equipped with brakes which prevent the manipulator from collapsing when power is removed from the servomotors. In "free" mode, power is supplied to the joint encoders but no power is supplied to the joint motors and all brakes are released.

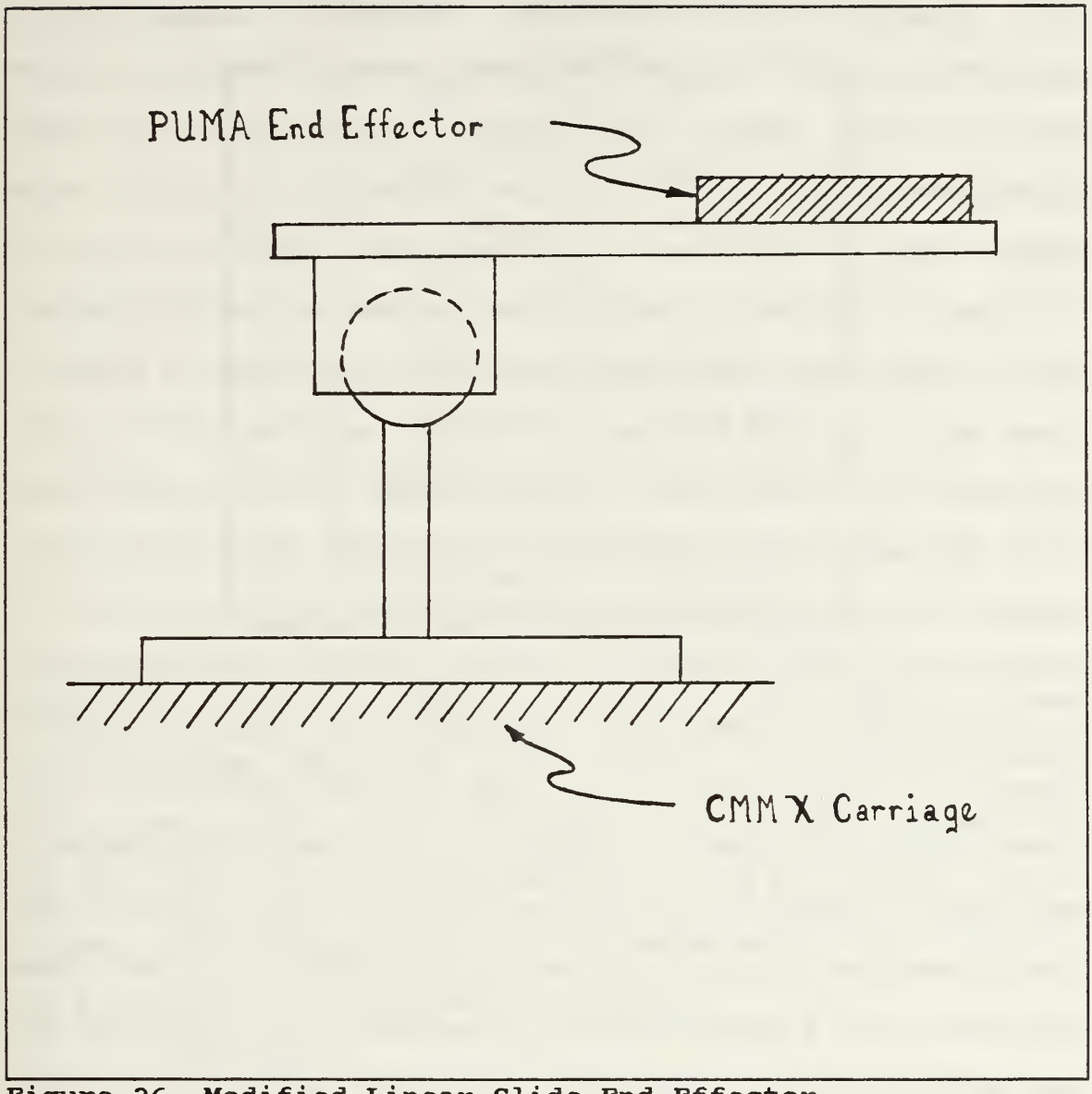

Figure 26. Modified Linear Slide End Effector

rotation about any set of three orthogonal axes placed at the center of the ball. Ball joints are fixed in translation. Arbitrary rotational motion and the fixed translation relationship as observed through a series of measurements will define a point in space, but no set of fixed axes can be associated with this point. Therefore, F^E is assigned to the point in space defined by the ball joint. Note that the ball

joint must be "exercised" in all its degrees of freedom as observed through a series of measurements in order for the model to hold. Now as the point defined by the ball joint translates along the axis of the CMM base, a specific axis rather than a direction is defined. The location of F^M is arbitrarily defined by zeroing the readout on the CMM display unit at some point along the axis. This zero point is fixed in space with respect to the ball joint location. F^M is then defined to be this fixed point in space and the axis along which the ball joint travels is associated with F^M as well. Summarizing, F^M consists of a fixed point and an axis, and F^E consists of simply a point. As noted earlier, any three of the six parameters of RPYT are sufficient to transform from a frame to a point as long as one of the parameters is a translation. The rotation $Rot(\phi,z_6)$ and translations along x and z were chosen for T_5^E. Since F^M is not fully defined, the transformation T_M^0 is developed by considering T_0^M and then inverting this transformation. In general, two rotations are required to align a particular frame axis with an arbitrary axis in space. Three more parameters are then required to transform to a point on the axis. The kinematic parameters for T_M^0 and T_5^E are listed in Table 5 where the identifiable parameters are typed in bold.

With these eight identifiable parameters added to the 18 identifiable parameters of the PUMA kinematic model, the closed chain model consists of a total of 26 identifiable

TABLE 5. KINEMATIC PARAMETERS FOR T_M^O AND T_5^E

	T_M^O	T_5^E
ϕ	195.0°	-41.0°
θ	-30.0°	0.0°
ψ	0.0°	0.0°
x	-180.0 mm	78.0 mm
y	-380.0 mm	0.0 mm
z	360.0 mm	79.0 mm

parameters. With the PUMA end effector ball joint at some position along the slide, the x, y and z position is known with respect to F^M. The x coordinate is the displacement of the ball joint from the zero reference, and the y and z values are zero. Therefore, a minimum of 9 measurements are necessary for a solution in the absence of noise.

4. Simulation

a. Introduction

In the CMM experiment, simulation data was easily developed by generating a random set of joint variables and computing the forward kinematic solution since the pose of the end effector was not important. However, in this experiment, for a given x coordinate, the end effector position is known and the joint variable angles corresponding to that position

must be determined. This is a special case of the so called inverse kinematic solution. In general, there is no guarantee that an analytic inverse solution exists. Furthermore, there are problems of uniqueness and numerical instabilities that arise from the cosecant and secant functions that are inherent in the analytic expressions. Consequently, a numerical approach is used in this case to compute the simulation data.

The position data can be generated by randomly generating a value for x displacement along the slide. As stated earlier, the y and z components are zero. These values become the "desired" end effector position. For a given set of joint variables, a forward solution based on the given model can be calculated resulting in an end effector position. The difference between the calculated end effector position and the "desired" position, becomes a set of error functions to be minimized. The problem statement in a form suitable for ZXSSQ implementation is:

- Minimize: $(f_1(\mathbf{x}))^2 + (f_2(\mathbf{x}))^2 + (f_3(\mathbf{x}))^2$
- Over: $\mathbf{x} = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$

where f_1 is the difference between the calculated and "desired" x coordinate of the end effector, and f_2 and f_3 are the calculated y and z coordinates.

The suite of simulation programs and their interaction is similar to the CMM simulation scheme and is illustrated in Figure 27. Program listings are found in Appendices A, B and C.

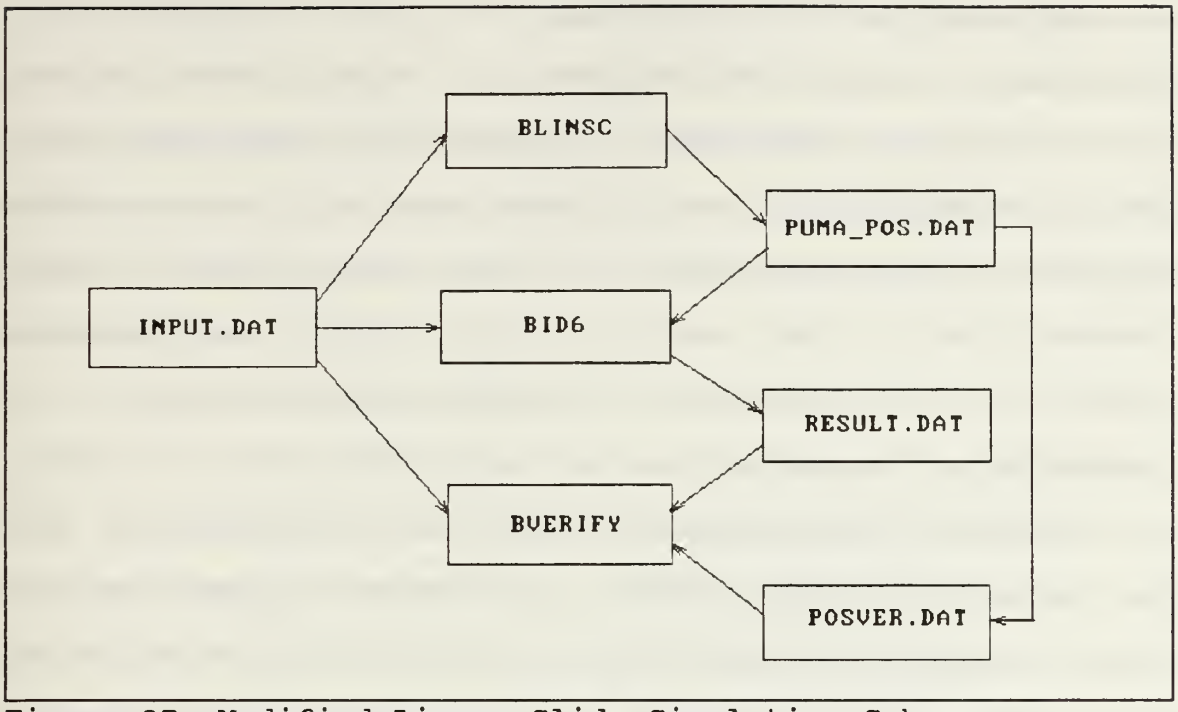

Figure 27. Modified Linear Slide Simulation Scheme

b. The Program BLINSC

BLINSC reads in the kinematic parameter table and additional information as described in the CMM experiment from a file titled INPUT.DAT. Known length and angular offsets are added to the parameters. The **x** vector described in the preceding paragraph is initialized. The **x** vector can be initialized to all zeroes or may be initialized such that solutions are "driven" to a particular arm configuration.[3] The program then generates a random value of the slide position,

---

[3]Referring back to Figure 16, the base of the manipulator is sometimes referred to as the waist and Joint 1 retation would be at the waist. Joint 2 then corresponds to a shoulder joint. Joint 3 is referred to as the elbow joint. With this terminology, when viewing the robot, four possible arm configurations are possible: "lefty-elbow up", "lefty-elbow down", "righty elbow up", and "righty-elbow down". The manipulator is shown in "lefty-elbow down configuration in Figure 24.

x coordinate of the end effector, and the ZXSSQ option parameters are assigned appropriate values. ZXSSQ is then called to compute the inverse kinematic solution as described in the preceding section. A modified version of the subroutine PUMA_ARM utilized previously in the program CID6 is then incorporated in the program as the ZXSSQ user supplied subroutine for calculation of the forward solution and evaluation of the error functions. Upon termination of ZXSSQ, the program provides capability for noise injection on both the three coordinates and the joint variables. The simulated joint variables and end effector coordinates are then written to a file titled PUMA_POS.DAT. The preceding process is repeated until the desired amount of simulated data has been generated.

c. The Program BID6

The only significant modification to previously described versions of ID6 is to the ZXSSQ user supplied subroutine PUMA_ARM. Additionally, modifications due to the changes in the number of identifiable parameters of the model, such as the size and makeup of the **x** vector, are incorporated into BID6. The modification to PUMA_ARM is similar to that used in BLINSC. Since only the position of F^E is identifiable, only three error functions can be calculated for each measurement set which, in this case, is the measured x, y and z position and the associated joint variables for that position. The error functions are the difference between the

calculated position of F^E based on the current value of all parameters and joint variables and the measured position. After execution of ZXSSQ, the calibrated parameters are written to a file RESULT.DAT as before. Additionally, the known position error only is calculated and written to RESULT.DAT.

d. The Program BVERIFY

As a final test of the accuracy of this method, the program VERIFY described for the CMM experiment was modified. BVERIFY computes a forward solution based on the actual model parameters and the calibrated parameters, and compares the resulting poses. To compute the actual pose, BVERIFY reads file INPUT.DAT and, as before, adds the length and angular offsets to the appropriate kinematic parameters. The calibrated kinematic parameter table is read from the file RESULT.DAT. Two separate forward solutions are calculated based on the two sets of kinematic parameters and sets of random joint angles produced by BLINSC and read from a renamed file POSVER.DAT. An average position error is calculated from the difference between the position entries of the two $\mathbf{T}^E$ matrices resulting from the forward kinematic solution calculations.

5. Experiment

a. Data Acquisition

With the CMM base rigidly affixed to the ramp and the ramp secured to the table, the upper ball joint flange is

bolted to the PUMA end effector flange and the end effector positioned above the slide by use of the teach pendant. With one operator supporting the PUMA, a second operator places the manipulator in "free" mode. The manipulator is then maneuvered into position so that the lower ball joint flange can be bolted to the X carriage of the CMM. Once the lower flange is bolted, the carriage is placed at a predetermined location and the display unit readout zeroed. This location with respect to approximate location of the base or zero frame of the manipulator is measured and recorded for incorporation into the kinematic parameter table along with the ramp orientation and the ball joint position and orientation with respect to frame 5 of the manipulator.

At the zero position and nine other positions located at approximately 80 mm intervals, four separate sets of measurements were recorded. The position for each set remained fixed by tightening the provided thumbscrew onto one of the guide bars. The manipulator was then maneuvered into four widely varying configurations and the joint variables for each configuration recorded. After completion of the 40 measurements, the end effector was unbolted and then joint one was rotated to the other arm configuration. A second set of measurements in the second arm configuration were recorded.

b. Parameter Identification and Verification

The entire set of 80 measurements were then used by an experimental version of BID6 for actual parameter

identification. Table 6 lists the nominal and calibrated parameters for this method. As before, length and angular parameters are reported in units of milli-meters and degrees, respectively. As in the CMM experiment, the data was then split into two groups for verification. The resulting positional error was 0.744 mm.

TABLE 6. NOMINAL AND CALIBRATED KINEMATIC PARAMETERS

	Nominal	Calibrated
ϕ_M	195.0	194.903
θ_M	-90.0	-30.887
x_M	-180.0	-179.567
y_M	-380.0	-378.528
z_M	360.0	355.635
a_1	0.0	-0.096
α_4	-90.0	-89.823
$\delta\theta_2$	0.0	-0.340
a_2	431.85	431.123
α_2	0.0	0.580
β_2	0.0	0.485
$\delta\theta_3$	0.0	-0.993
a_3	149.09	146.028
a_3	-20.33	-20.255
a_3	90.0	90.415
$\delta\theta_4$	0.0	-1.089
a_3	433.0	434.095
a_4	0.0	0.074
α_4	-90.0	-90.244
$\delta\theta_5$	0.0	1.293
d_5	0.0	-0.863
a_5	0.0	-0.175
α_5	90.0	89.905
ϕ_E	-41.0	-41.355
x_E	78.0	78.404
z_E	79.0	79.203

C. THE WIRE POTENTIOMETER METHOD

1. Introduction

Calibration of PUMA 560 was performed by Driels [Ref. 12] utilizing an instrument referred to as a ball bar. The instrument consisted of a rigid bar of known fixed length with a ball joint attached at each end. One end of the ball bar was affixed to a work surface within the working volume of the PUMA. The other ball joint was attached to the end effector flange of the robot. The method offered several advantages including:

- precise length measurements
- essentially no measurement noise
- ease of fabrication
- low cost

The major disadvantages with the method are:

- Calibration can only be performed on a manipulator with a "free" mode of operation.
- The method requires at least two operators so that the manipulator is supported while collecting data.
- The end effector, and hence data collection, is limited to the surface of a sphere of ball bar length radius which may limit its applicability in certain environments.

The wire potentiometer method was developed to retain the advantages of the ball bar method while overcoming the disadvantages which arise from constraining the manipulator end effector.

2. Physical Description of the Measurement System

a. The Wire Potentiometer

The wire potentiometer used in this experiment is a Celesco model, DV301-0050-111-III0, and is illustrated in Figure 28. This model is designed to produce linear displacement and velocity measurements with a maximum travel of 50 inches. The calibration process is only concerned with displacement measurements which are transduced by means of a proportional resistance from the wiper arm of 0-500 Ω potentiometer.

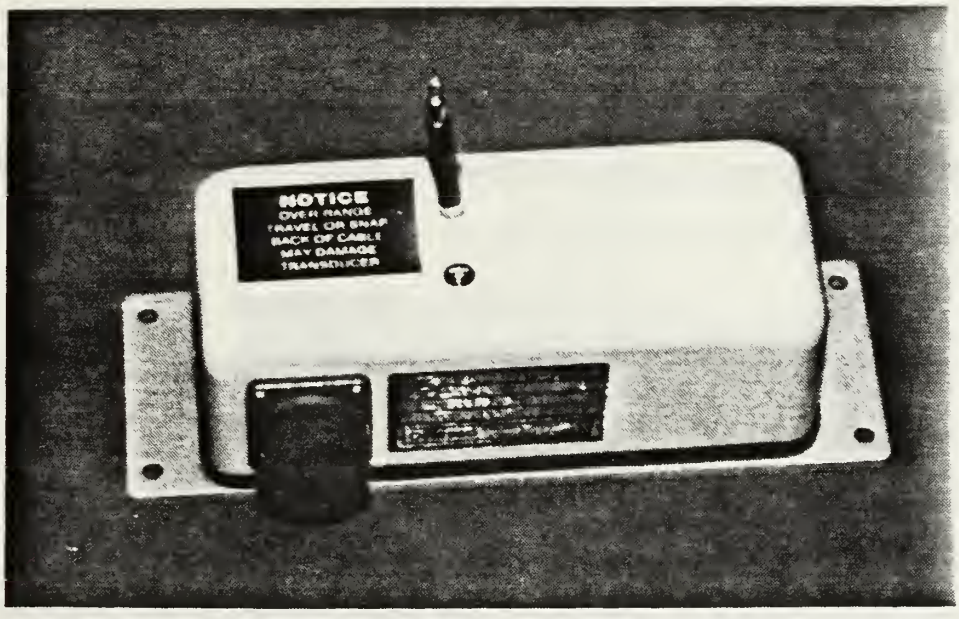

Figure 28. Celesco Wire Potentiometer

b. Fixture Design for Measurements in a Volume

As noted above, the wire potentiometer was designed for measurement of linear motion. To effectively

utilize this device in a calibration application, it was necessary to design a fixture capable of measuring distances within a working volume. Several factors were considered when designing the fixtures:

- simplicity
- low measurement noise
- prevention of wear and deformation of the wire
- mathematic modelling

The fixture designs for the measurement system base and end effector are illustrated in Figures 29 and 30, respectively.

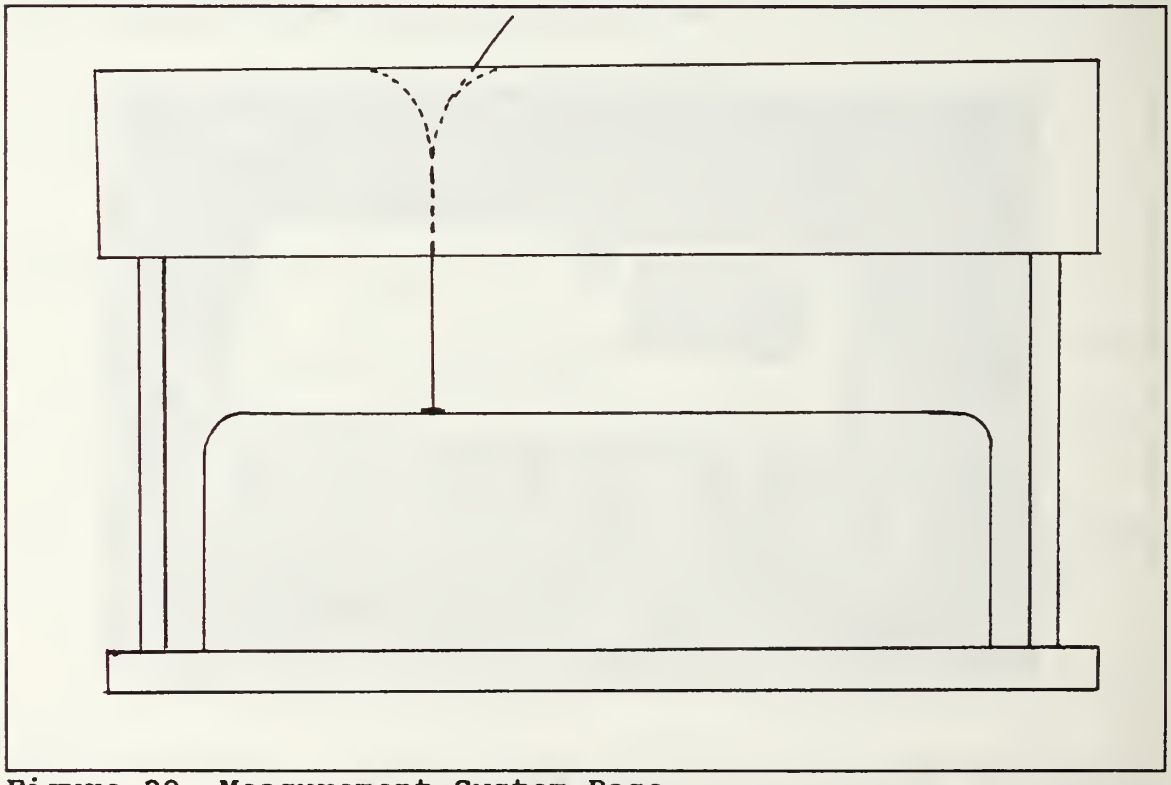

Figure 29. Measurement System Base

All parts were constructed from aluminum. The funnel shaped ports were cut with a carbide tipped half-inch radius beading router bit. The known radius of curvature of

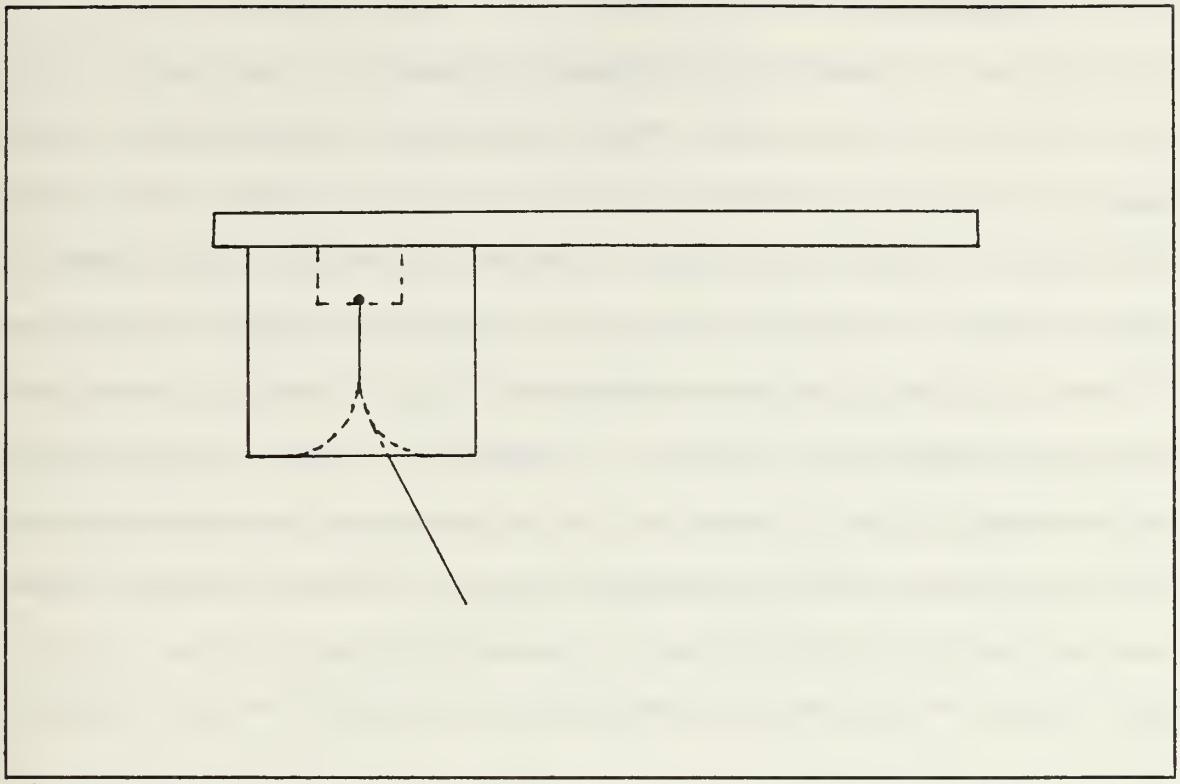

Figure 30. End Effector Fixture

the funnel surface can easily be incorporated into a mathematical model, as well as reduce the possibility of wire deformation. The funnel surfaces are highly polished, which reduces both abrasive wear on the wire and measurement noise by allowing the wire under tension to align itself in a minimum length configuration. A detailed description of the geometry and wire length calculation is provided in the following section. A tradeoff between smooth travel and minimal play at the throat of the funnel was made when determining the internal passage diameter.

3. Theory

a. Closed Chain Kinematic Model

As in the two preceding experiments, the manipulator kinematic parameters are the same as those listed in Table 1. The measurement system kinematic model must be developed so that the makeup of F^M and F^E are known which then fixes the identifiable parameters in T_M^O and T_5^E. If the wire had been "perfectly flexible in bending", then instead of the funnel shaped port, the wire could have been passed through a fixture with a port of essentially equal diameter as the wire, as illustrated in the planar view of Figure 31.

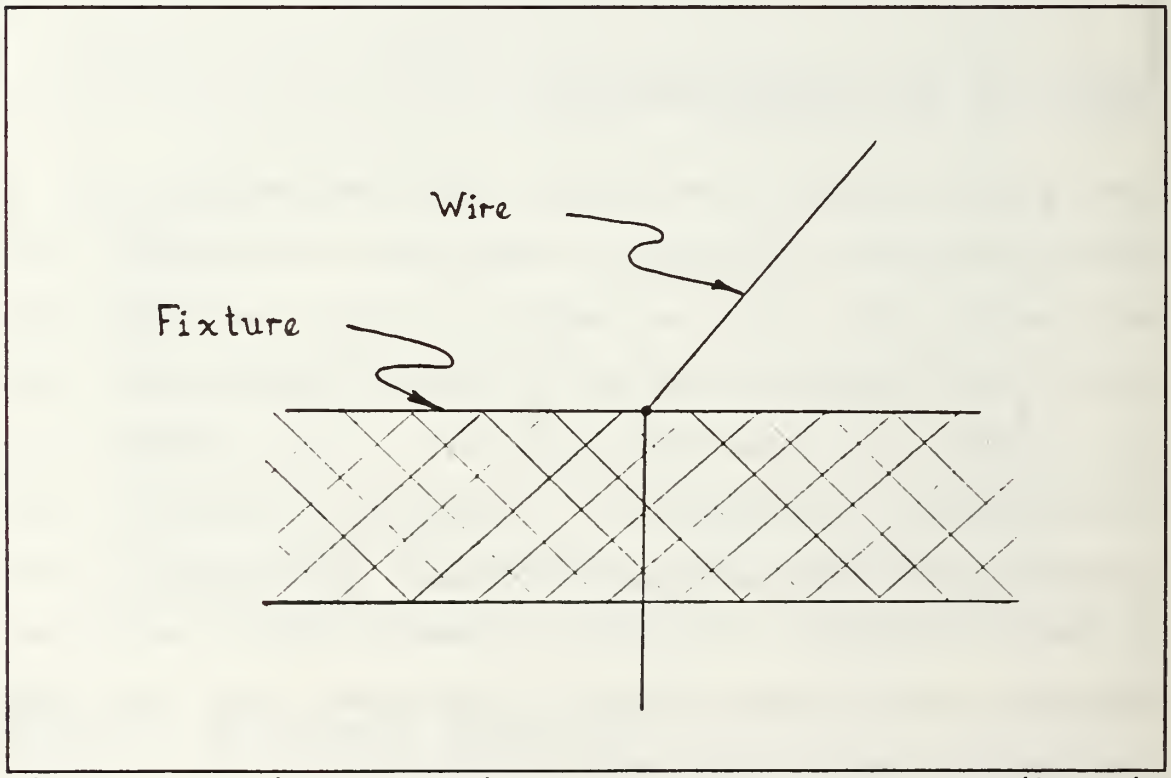

Figure 31. Fixture Design for a "Perfectly Flexible in Bending" Wire

In this case, the model resembles the ball joint model described previously. Point O in Figure 31 is fixed in translation but the wire is "free" to assume any orientation above the fixture. Point O could then be defined as the measurement system reference point. With a similar fixture mounted on the end effector flange of the PUMA, a second point would be defined. This point becomes the origin of F^E. Summarizing, F^M and F^E consist of well defined points in space but no axes can be associated with the measurement system. The same concepts hold for the funnel shaped ports with the defined point now located at the throat of the funnel. However, due to the axial symmetry of the funnel, an axis can be associated with the F^M and F^E as illustrated in Figure 32. This theoretically identifiable axis was not included in the model used in this experiment based on the following assumptions:

- The z axis illustrated in Figure 31 for the measurement system base and end effector fixtures would be nominally aligned with the z_o and z_5 axis respectively.
- The small radius of curvature would make identification of any nonalignment unreliable.
- Any errors induced would be sufficiently small that they would not be discernible from normal measurement noise.

Without the axes included in the model, three parameters are identifiable in both $\mathbf{T}_M^o$ and $\mathbf{T}_5^E$. As stated before, any three of the six parameters in each transformation can be used as long as one parameter is a translation. The nominal values of the parameters chosen for identification are typed in bold in

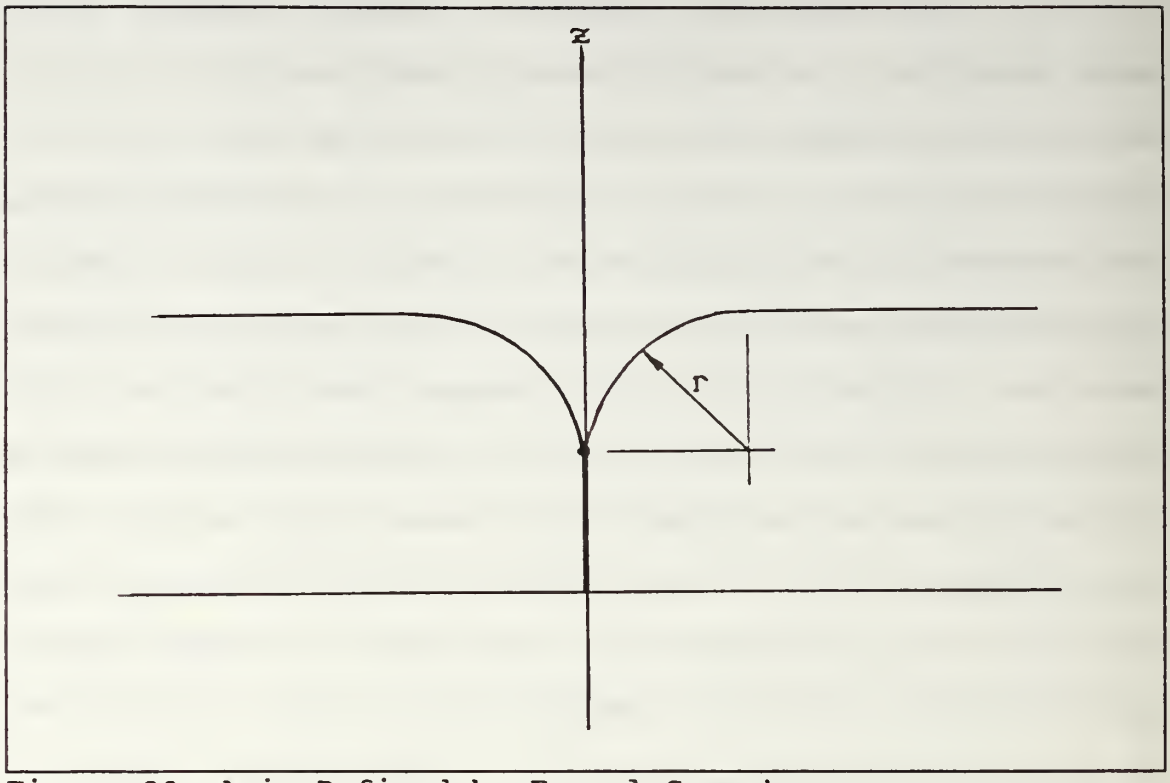

Figure 32. Axis Defined by Funnel Geometry

the kinematic parameter table, Table 7. Recall that although an independent set of axes are not defined at either F^M and F^E, by virtue of the transformations, a dependant set of axes is defined and can be referenced.

As developed, the total model consists of 24 identifiable parameters. For a given configuration of the manipulator, the measured quantity is the length of wire between the origins of F^M and F^E. Therefore, in the absence of noise, a minimum of 24 measurements are required to adequately define the problem.

TABLE 7. KINEMATIC PARAMETER TABLE FOR T_M^O AND T_5^E

	T_M^O	T_5^E
ϕ	0.0	-135.0
θ	0.0	0.0
ψ	0.0	0.0
x	-75.0	-76.2
y	-711.0	0.0
z	552.0	76.2

b. Calculation of Wire Length from Kinematic Parameters

During each iteration in the parameter identification algorithm, a "predicted" value of the wire length based on the current value of the parameters must be computed for comparison. The position and orientation of the end effector fixture reference frame with respect to the measurement frame is provided, as before, by the T^E matrix.

As noted earlier, the wire under tension will align itself in a minimum length configuration between the base fixture and the end effector fixture. In this configuration, the wire departs the funnel surfaces tangentially and in a direction such that the tangent to

tangent distance is minimum. Figure 33 illustrates the wire configuration for an arbitrary manipulator pose.

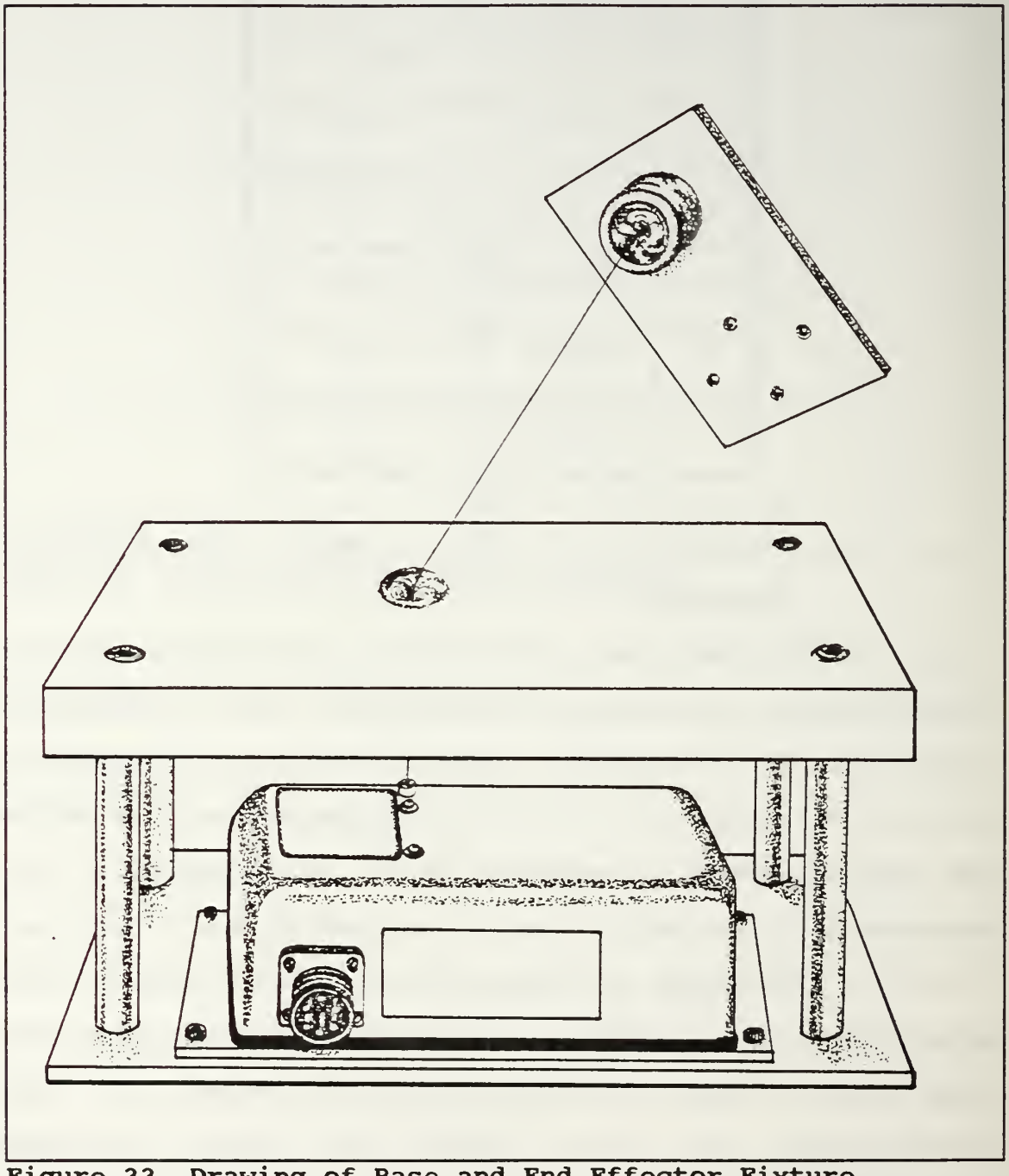

Figure 33. Drawing of Base and End Effector Fixture

Let M and E designate the origins of the measurement and end effector frames respectively. Define TM

and TE as the points of departure of the wire from the measurement base fixture and end effector fixture respectively. Note that the wire departs at points TM and TE tangentially from the funnel surface and will be referred to as the tangent points for the remainder of this discussion. Additionally, define the z axes for F^M and F^E to lie along the axis of the funnel as shown previously in Figure 32. The complexity of describing the wire path can be simplified somewhat by noting the following feature. If the wire path is projected onto the x-y plane of F^M and F^E then, as is shown in Figure 34, the projected paths M-TM-TE and E-TE-TM in each frame's x-y plane is a straight line. Only in unique configurations, namely each frame's z axes intersect, will the projected path M-TM-TE-E form a straight line.

The geometry described in this paragraph is identical for either frame so without loss of generality F^M is described. Let v define the axis in the xy plane of F^M which describes the line of action of the wire path projection as shown in Figure 34. Figure 35 illustrates the geometry as viewed in the vz plane of the frame. If the x and y coordinates of the tangent point TM (tm_x and tm_y) in Figure 35 are known, then tm_z is fixed by the known radius of curvature. Furthermore, the angle θ can be determined which then results in a solution for arc length O-TM. Additionally, the angle ϕ can be calculated which then fully defines the direction of line segment TM-TE. From this analysis, it is clear that the

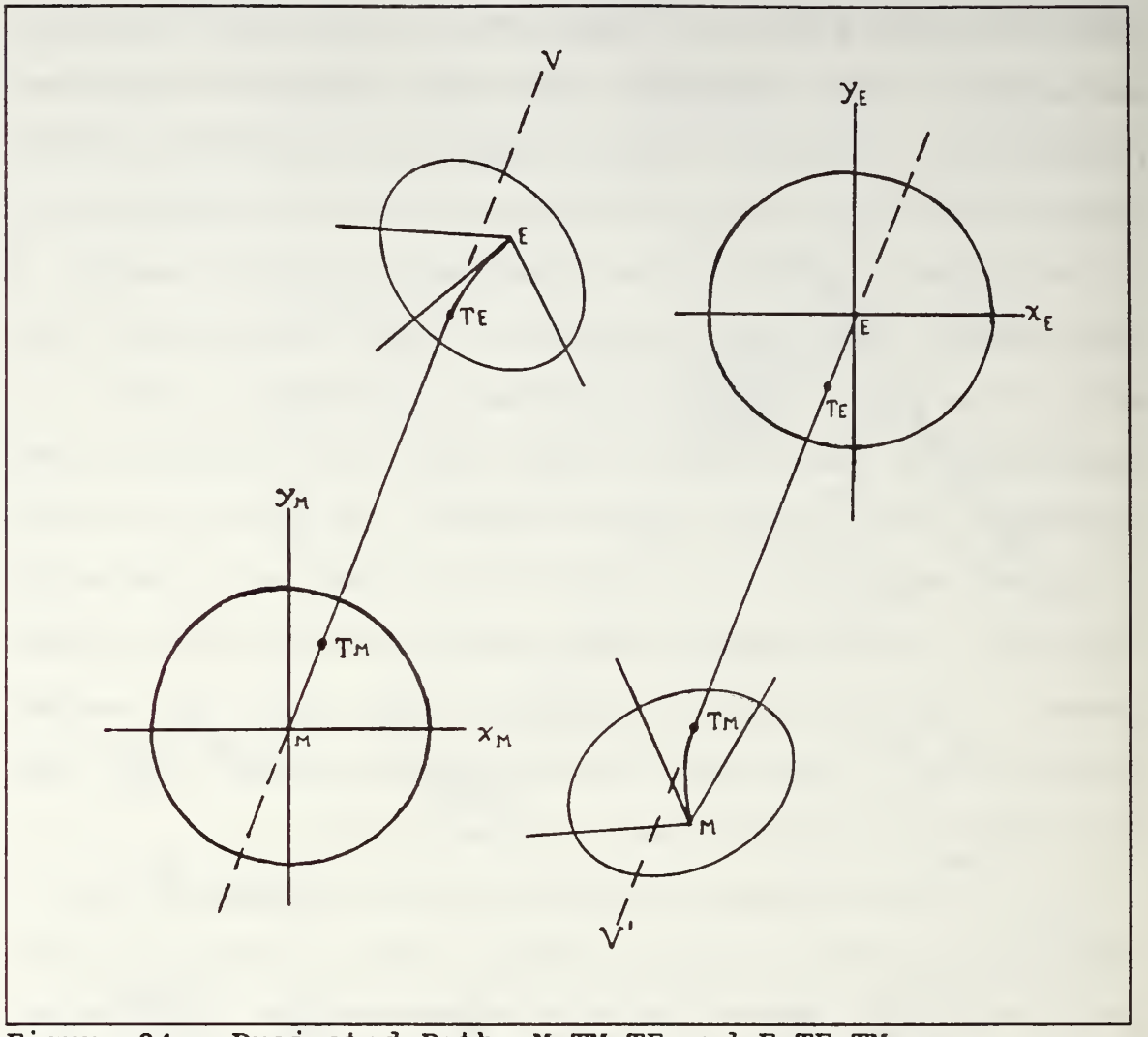

Figure 34. Projected Paths M-TM-TE and E-TE-TM

wire length solution is a function of the x and y components of the tangent points of both fixtures.

$$L = L\ (tm_x,\ tm_y,\ te_x,\ te_y) \qquad (43)$$

Although an analytical solution to this problem most likely exists, it would clearly reduce to solving 4 non-linear equations with the 4 unknowns ultimately requiring some numerical technique to obtain a solution. Consequently, an optimization scheme was chosen for the solution method at the

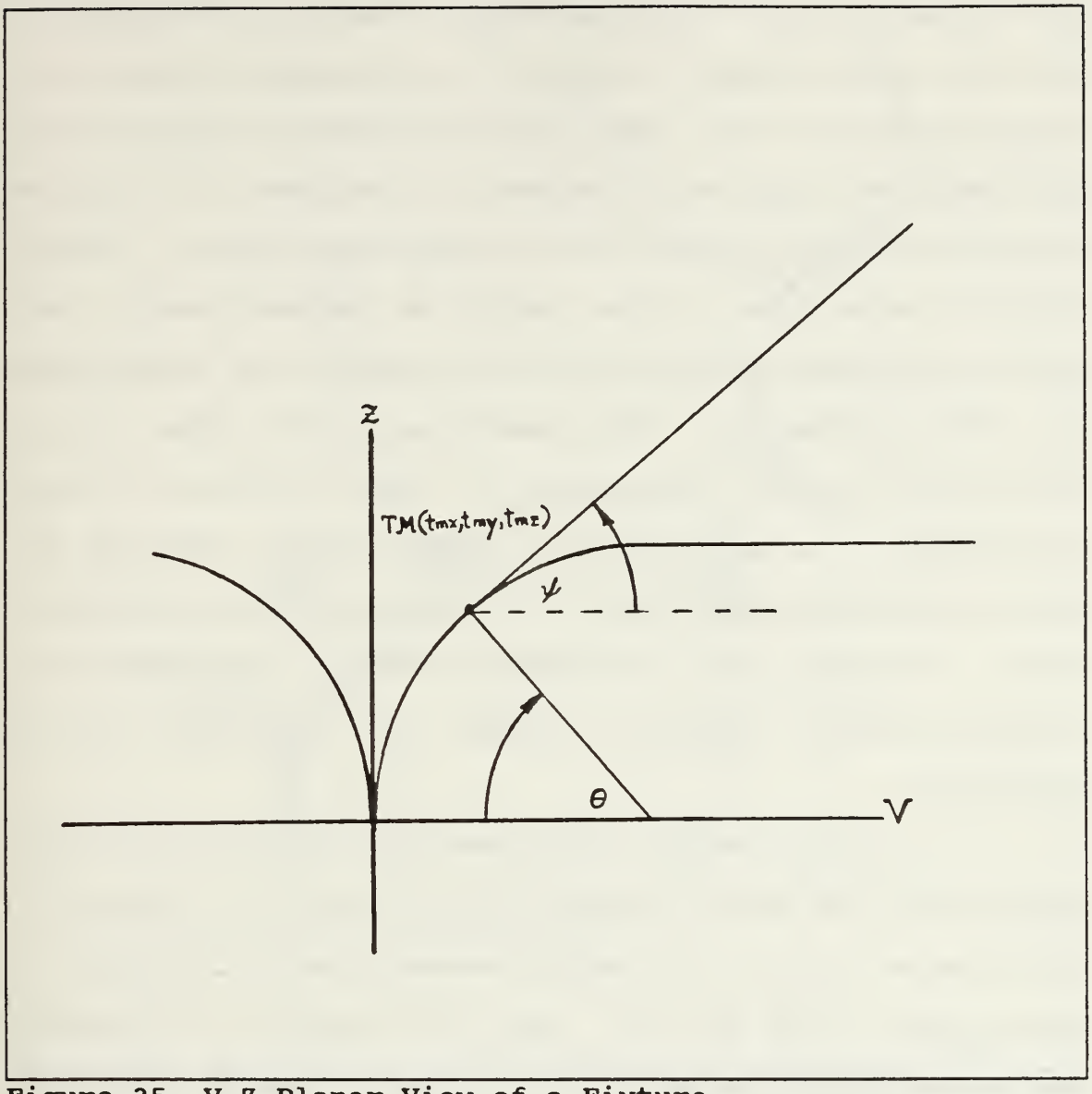

Figure 35. V-Z Planar View of a Fixture

outset. The general approach was to develop a unit vector for the tangential direction based on a variable x and y position in each fixture's reference frame. The unit vector calculated in the end effectors reference frame is then transformed into measurement frame coordinates. These vectors are designated $\mathbf{u}_M$ and $\mathbf{u}_E$ and their component by component sum will be zero when the wire is in a minimum length configuration. The three sums

form three functions to be minimized. However, as illustrated with the simple example in Figure 36, alignment of these unit tangent vectors alone does not sufficiently constrain the problem. The additional constraint required is that the tangent to tangent length TM-TE be minimized as well. A fourth function consisting of the tangent to tangent length could then be included for minimization. However, as stated thus far, the problem is not proportional with three of the functions ideally converging to zero, and the fourth converging to a comparatively larger number. Scaling the length function could reduce this problem to an acceptable level. However, the following method eliminates all disproportionate aspects between the functions to be minimized.

Let $\mathbf{u}_{tt}$ be the vector describing the line of action between TM and TE as illustrated previously in Figure 36. Calculation of $\mathbf{u}_{tt}$ is easily accomplished utilizing the coordinates of TM_M and TE_M. Now the component by component difference between $\mathbf{u}_M$ and $\mathbf{u}_{tt}$ will be zero when the wire is in a minimum length configuration. These component by component differences can then form three additional functions to be minimized. Note that the difference between $\mathbf{u}_E$ and $\mathbf{u}_{tt}$ would have worked equally well. In normal problem statement form:

- Minimize: $\Sigma(f_i(\mathbf{x}))^2 \quad i=1,2,\cdots,6$
- Over: $\mathbf{x} = [{}^E x_E, {}^E y_E, {}^M x_M, {}^M y_M]$
- f_1, f_2 and f_3 are the component by component sum of $\mathbf{u}_M$ and $\mathbf{u}_E$

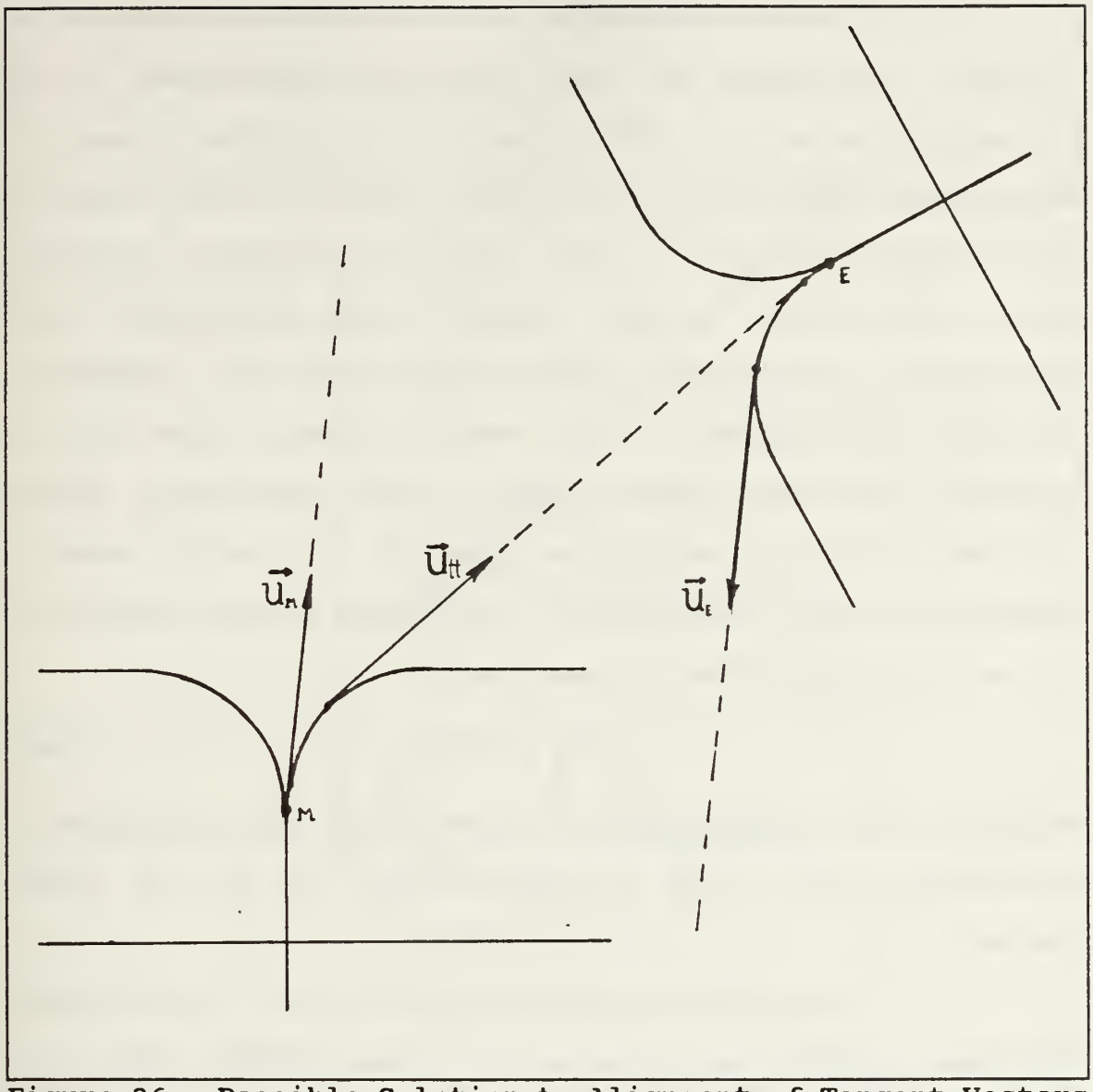

Figure 36. Possible Solution to Alignment of Tangent Vectors

- f_4, f_5 and f_6 are the component by component difference between u_M and u_{tt}
- Recall that the superscripts on the elements of **x** refer to which frame the variables are associated with and the subscript refers to which frame their numerical values are referenced to.

With a numerical solution obtained, the wire length can easily be computed from the components of **x**.

To aid convergence, a good initial estimate of the x and y coordinates for each frame are determined in the following manner. A unit vector **v** directed from the measurement frame origin to the end effector frame origin is calculated from the x, y and z position elements of the T^E matrix. The initial x and y values in the measurement frame are chosen as the funnel radius times the i and j components of this unit vector. A unit vector directed from the end effector reference frame origin to the measurement system reference origin is simply **-v**. However, this vector must be described in the end effector's reference frame. Calculation of $-\mathbf{v}_E$ is accomplished by Equation 44.

$$-\vec{V}_E = (\boldsymbol{T}^{\boldsymbol{E}})^{-1} (-\vec{V}_M) \qquad (44)$$

As in F^M, the initial x and y values in F^E are calculated by multiplying the i and j components of $-\mathbf{v}_E$ by the funnel radius.

Given the initial or updated x and y coordinates in frame coordinates, calculation of the tangent vector with respect to that frame proceeds as follows and is identical for both frames. As before, define an axis v that lies in the frames x-y plane and intersects the point defined by the current values of x and y and the frames origin. Figure 37 is a view of the plane formed by the v and z axis. Point P has coordinates (x,y,0) and point O is the frame origin. The distance from O to P is found from Equation 45.

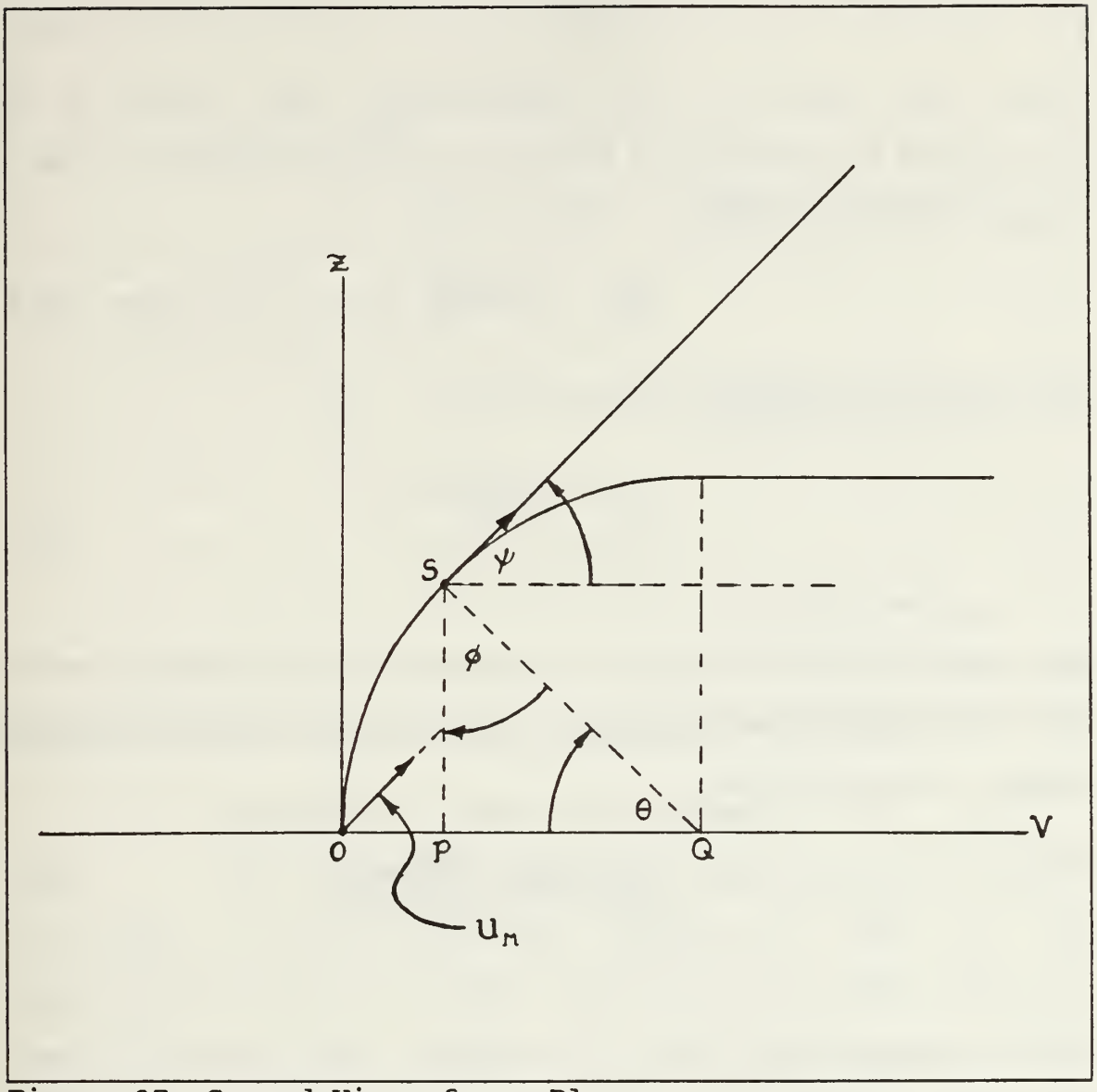

Figure 37. Second View of v-z Plane

$$\overline{OP} = \sqrt{x^2+y^2} \tag{45}$$

The length of line segment OQ is equal to the radius, r, of the funnel. Calculation of the side PQ of the right triangle PQS is accomplished with Equation 46.

$$\overline{PQ} = r - \overline{OP} \tag{46}$$

Since the length of SQ is the radius, then length SP is calculated by Equation 47 and this value corresponds to the z component of the current tangent point.

$$\overline{SP} = \sqrt{r^2 - \overline{PQ}^2} \tag{47}$$

The angles ϕ and θ are calculated by

$$\begin{aligned} \phi &= \sin^{-1}\left(\frac{\overline{PQ}}{r}\right) \\ \theta &= \sin^{-1}\left(\frac{\overline{PS}}{r}\right) \end{aligned} \tag{48}$$

The angle θ will be used at convergence to calculate arclength OS. The unit tangent vector, either $\mathbf{u}_M$ or $\mathbf{u}_E$ is calculated as shown in Equation 49.

$$\boldsymbol{u} = \frac{x\,\boldsymbol{i} + y\,\boldsymbol{j} + \overline{OP}\tan\phi\,\boldsymbol{k}}{\sqrt{x^2 + y^2 + (\overline{OP}\tan\phi)^2}} \tag{49}$$

4. Simulation

a. Introduction

The general simulation scheme is similar to the preceding calibration methods. A flowchart of the scheme is illustrated in Figure 38. The programs are listed in Appendices D, E and F. Prior to a description of the main programs in the simulation process, the subroutines LENGTH and MINLENTH which calculate the wire length will be discussed.

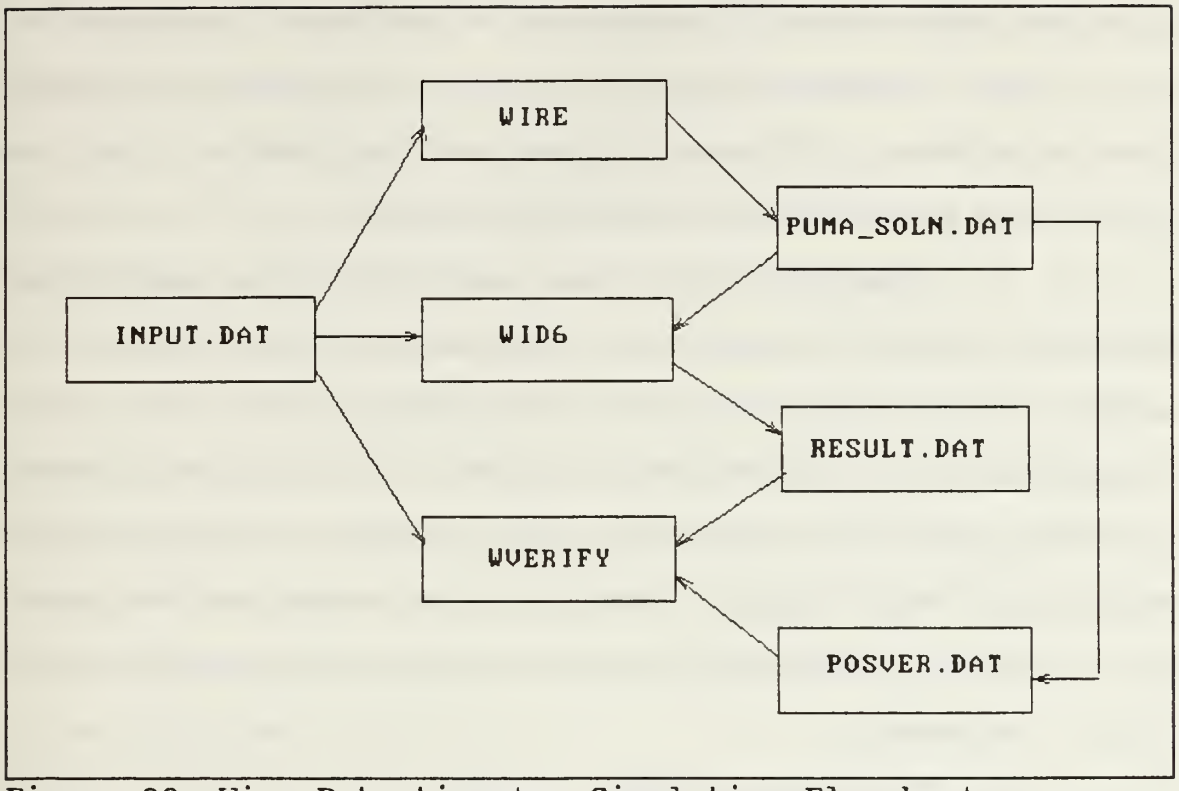

Figure 38. Wire Potentiometer Simulation Flowchart

b. Subroutines LENGTH and MINLENTH

The subroutine LENGTH and its subroutine MINLENTH are used in each simulation program as well as the experimental version of WID6 and the program COMP which is used to verify the accuracy of experimentally calibrated parameters. The purpose of LENGTH is to calculate the wire length based on the $\mathbf{T}^E$ matrix supplied by the calling program.

LENGTH initializes the ZXSSQ option parameters and calculates $\mathbf{x}_o$, the ZXSSQ variable vector, from the current frame origins as described earlier. Upon termination of ZXSSQ, the overall wire length is calculated and passed back to the calling program. In program WID6, LENGTH is called from the subroutine PUMA_ARM which is similar to previously described

versions. Note that this represents an inner optimization loop. Since FORTRAN is not a recursive language, a renamed version of the ZXSSQ source code was required so that the inner loop optimization could be performed.

MINLENTH is the "user supplied subroutine" called by ZXSSQ. MINLENTH is passed the current value of **x**. The tangent vectors $\mathbf{u}_M$ and $\mathbf{u}_E$ are calculated as well as $\mathbf{u}_{tt}$. The coordinates of $\mathbf{u}_E$ are transformed into F^M coordinates and the six error functions are formed. The tangent to tangent length and the elevation angles θ which are necessary for the total wire length calculation are computed and passed to LENGTH upon ZXSSQ convergence.

c. The Program Wire

The function of the program WIRE is to generate simulated wire length and joint variable data. In a general sense, it performs similarly to the previously described programs JOINT and BLINSC. The nominal kinematic parameter table, along with length and angular offsets, length and angular noise levels, and the total number of simulated measurements to be generated, are read from the file INPUT.DAT. The length and angular offsets are added to the corresponding nominal parameters. These values will be used in forward kinematic solution calculations. Comparable to BLINSC, data simulation will require an inverse kinematic solution. Unlike BLINSC, the end effector pose must be constrained to realistically match the actual experiment. In other words, as

viewed from the end effector frame, the measurement system base frame must lie above the x-y plane of F^E.

The first step in the process is to establish the origin of F^E. This random point is developed using a spherical coordinate approach where random values of ϕ, θ and L, illustrated in Figure 39, are constructed from scaled values output from a Monte-Carlo random number generator. Note that ϕ was maintained in the range 0°-80° and L was maintained in the range of 100-1000 mm. The coordinates ϕ, θ and L are then converted to cartesian coordinates in the usual way. These three values become part of the "desired" result of the forward kinematic solution.

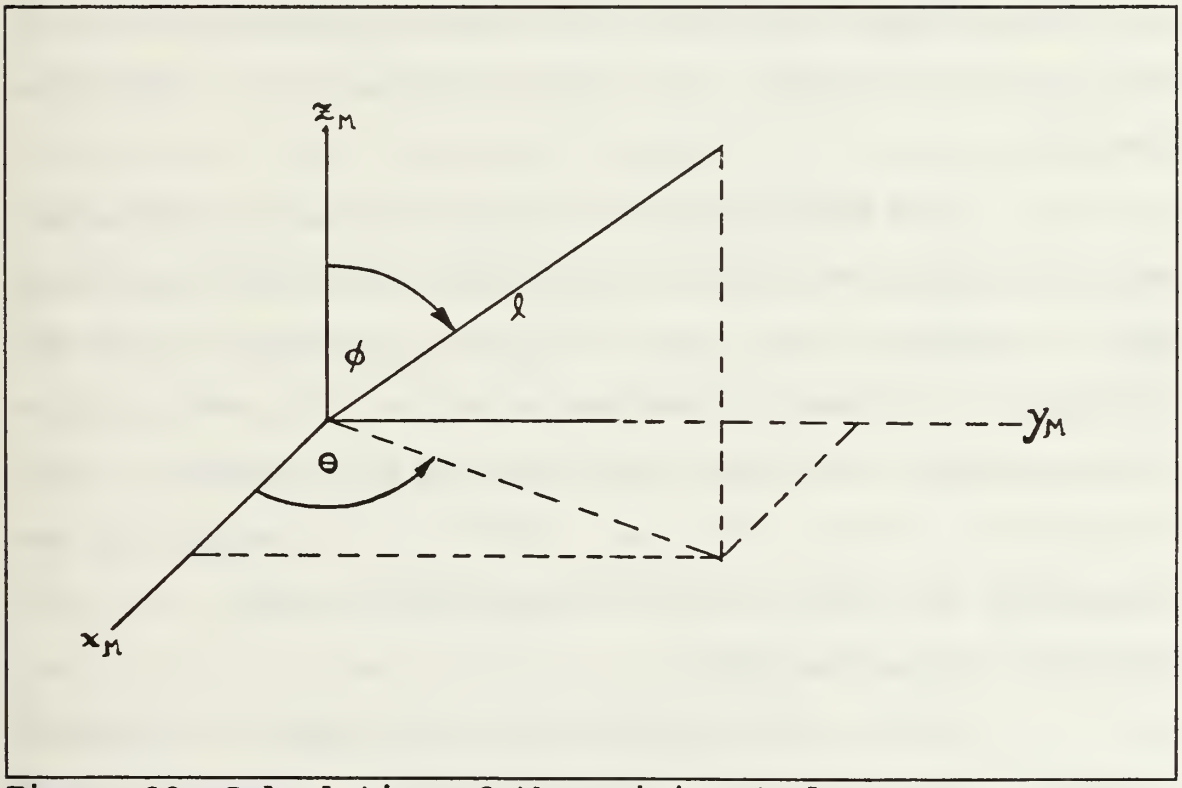

Figure 39. Calculation of the Origin of F^E

To ensure that the z axis of F^E was in a feasible direction, constrained random values of the z axis direction cosines were incorporated into the "desired" solution. This is accomplished by first calculating a unit vector **v** directed from the origin of F^E to the origin of F^M in F^M coordinates. Note that if these values were then used as the direction cosines for the z axis in T^E, then the z axis would "point" directly at the origin of F^M. These direction cosines are randomly adjusted about this value by a maximum range of ±30°. Denoting **u** as the unit vector made up of the perturbed direction cosines, feasibility is checked by computing the dot product of **u** and **v** which must be less than 90°. These six "desired" values correspond to the last two columns of the T^E matrix and are passed via common block to the subroutine PUMA_ARM.

The ZXSSQ vector x_o is then initialized along with the ZXSSQ option parameters. Subroutine PUMA_ARM is called by ZXSSQ to compute the error functions. As before, PUMA_ARM computes the forward kinematic solution, T^E, based on the current joint variables contained in the vector **x**. The difference between the upper right 3x2 elements of the calculated T^E matrix and the "desired" values form six functions to be minimized.

After ZXSSQ termination, the residual is checked to ensure validity of the calculated pose. If the solution is valid, the wire length is calculated by a call to subroutine

LENGTH. Random noise can be injected into the wire length and joint variables at this point. The wire length and joint variables are then written to a file PUMA_SOLN.DAT. The process is then repeated until the requisite number of simulated measurements have been generated.

d. The Program WID6

WID6 performs in a like manner to preceding versions of ID6. Minor changes have been made to reflect the differences in the identifiable parameters. The only significant difference is in the subroutine PUMA_ARM.

In this experiment, the measurement consists of the length of wire between F^M and F^E which is a function of the end effector pose and the fixture geometry. For each call to PUMA_ARM, a forward solution is calculated. With T^E calculated, the subroutine LENGTH is called and a corresponding wire length is calculated as described previously. The difference between the calculated wire length and the simulated data wire length form a single error function. This process is repeated until N error functions have been calculated where N is the number of measurements in the data set.

After ZXSSQ termination, the identified parameters are compared, as before, with the known parameters and these parameters and the rms position error are written to a file RESULT.DAT.

e. The Program WVERIFY

WVERIFY performs similarly to previous verification programs. The kinematic parameter table is read from the file RESULT.DAT. A second set of simulated joint variable and wire length data, generated by WIRE, is read from the file POSVER.DAT. A forward solution and then the corresponding wire length is calculated for each set of joint variables. The difference between the calculated lengths and the corresponding simulated length measurements is computed. An rms value for all errors is calculated and written to the terminal screen.

5. Experiment

a. Calibration of the Potentiometer

A model SE-2000 signal conditioner and digital display unit was provided with the wire potentiometer. However, the digital display was only capable of 0.01 inch accuracy which was inferior to the desired accuracy. The unit does provide a conditioned 0-10 volt analog output for displacement measurement. It was hoped that the potentiometer was capable of greater precision so it was calibrated in the following manner.

The end effector fixture was mounted in the chuck of a lathe and the base fixture mounted to the lathe carriage. The fixtures were aligned so that the wire was normal to both fixture upper surfaces as illustrated in Figure 40. The carriage was then positioned so that the upper fixture

surfaces were in light contact. The potentiometer was connected to the SE-2000 and the SE-2000 analog output was connected to a 5 1/2 digit DVM. The "zero" was adjusted on the SE-2000 for a zero volt reading on the DVM. Digital display, accurate to 0.001 mm, of lathe carriage travel is provided by an Acurite III display unit. Voltage and length readings were recorded at 50 mm intervals. Figure 41 is a Displacemnt vs Voltage plot of the data. As shown, a linear relationship results and Equation 50 is the linear best fit of the data. Figure 42 is a plot of the deviation of the data from the linear best fit over the length of the wire. The rms error in the deviation from Equation 50 is 0.22 mm. Although the potentiometer was not nearly as precise as desired, simulation with this level of noise still demonstrated robust convergence with the overall error approaching the accuracy of the measurement device which is less than the repeatability of the PUMA.

$$L = 0.12659V - 0.490203 \qquad (50)$$

b. Data Acquisition

A short data acquisition program was written to convert voltage measurements to millimeters using the linear relationship derived from the potentiometer calibration. This file then stored each measurement along with the corresponding joint variable data in a file titled PUMA_POS.DAT.

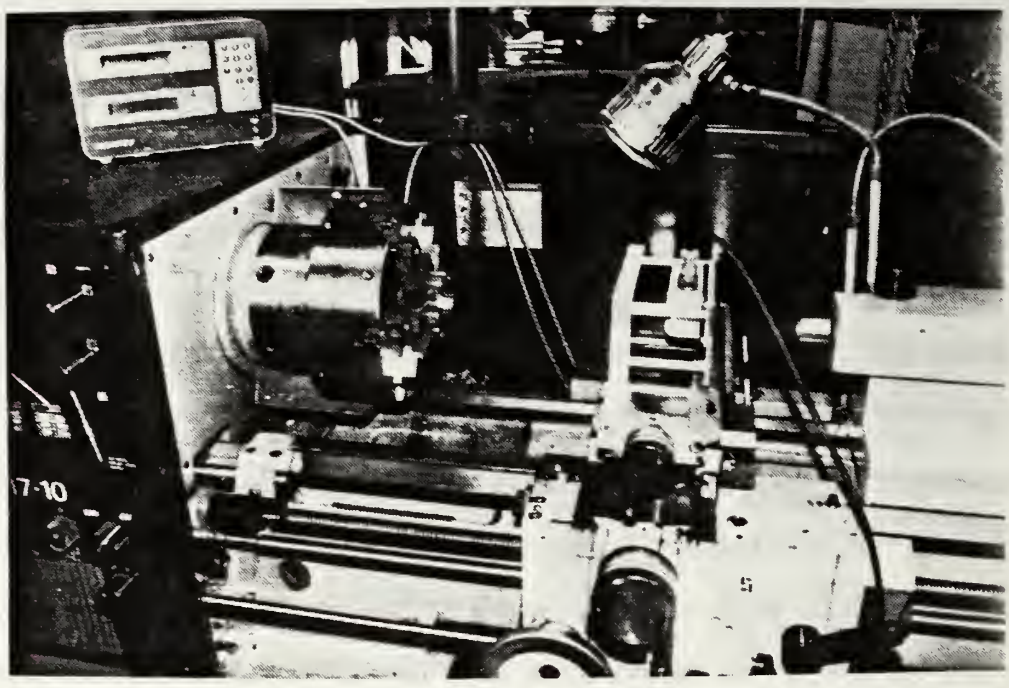

Figure 40. Calibration of the Potentiometer

The base fixture was secured to the work table as illustrated in Figure 43. As in the calibration setup, the potentiometer was connected to the SE-2000, and the analog output of that unit was connected to the 5 1/2 digit DVM. After sufficient equipment warm-up and with the end effector fixture resting on the base fixture, the voltage output was adjusted for a zero reading. The end effector fixture was then bolted to the PUMA. The PUMA was placed in a wide variety of joint variable configurations and data recorded. The length of the wire was sufficient to enable the PUMA to be placed in all four "arm configurations". 110 measurements were taken in approximately four hours by one operator entering data at a

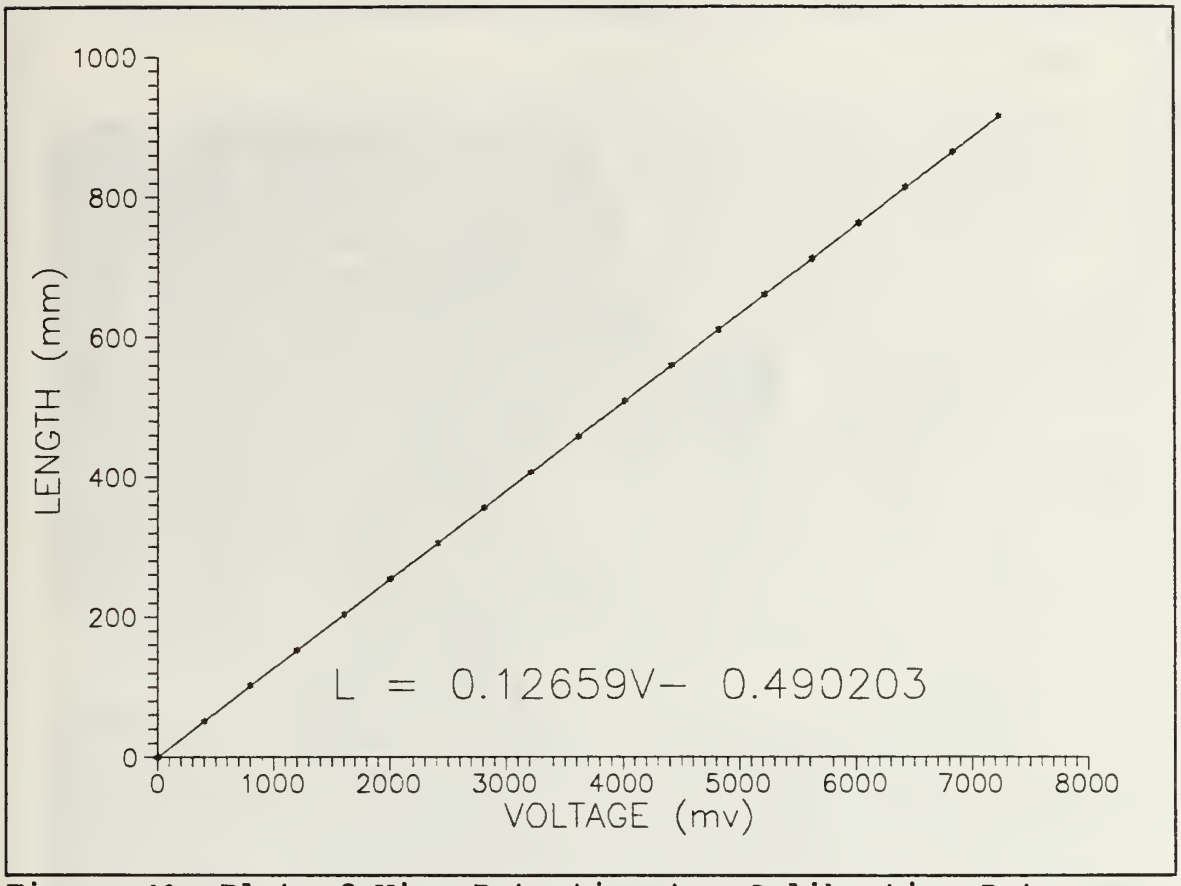

Figure 41. Plot of Wire Potentiometer Calibration Data

computer terminal and operating the PUMA with the teach pendant.

c. Parameter Identification and Verification

The entire data set was used for parameter identification in a version of WID6 modified for experimental data. The nominal and identified kinematic parameters are listed in Table 8 where the length and angular values are in units of millimeters and degrees, respectively.

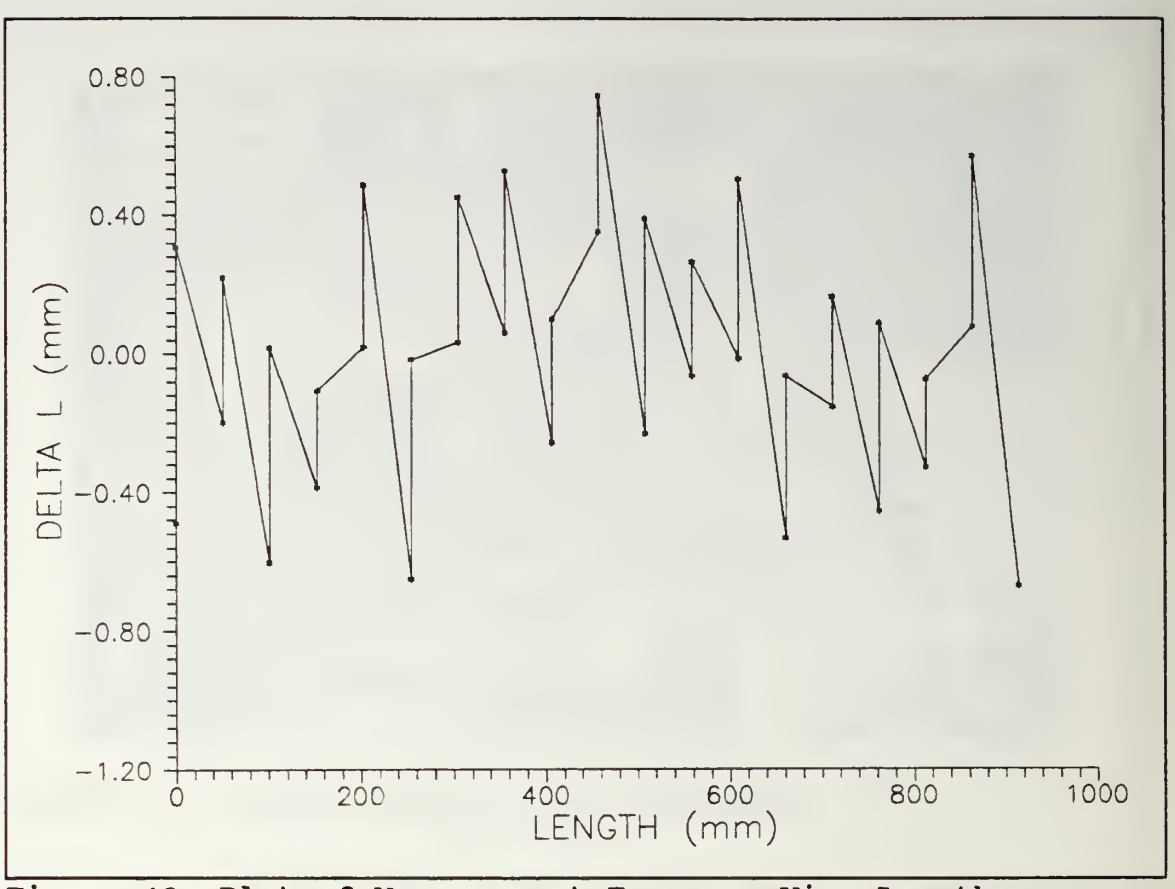

Figure 42. Plot of Measurement Error vs Wire Length

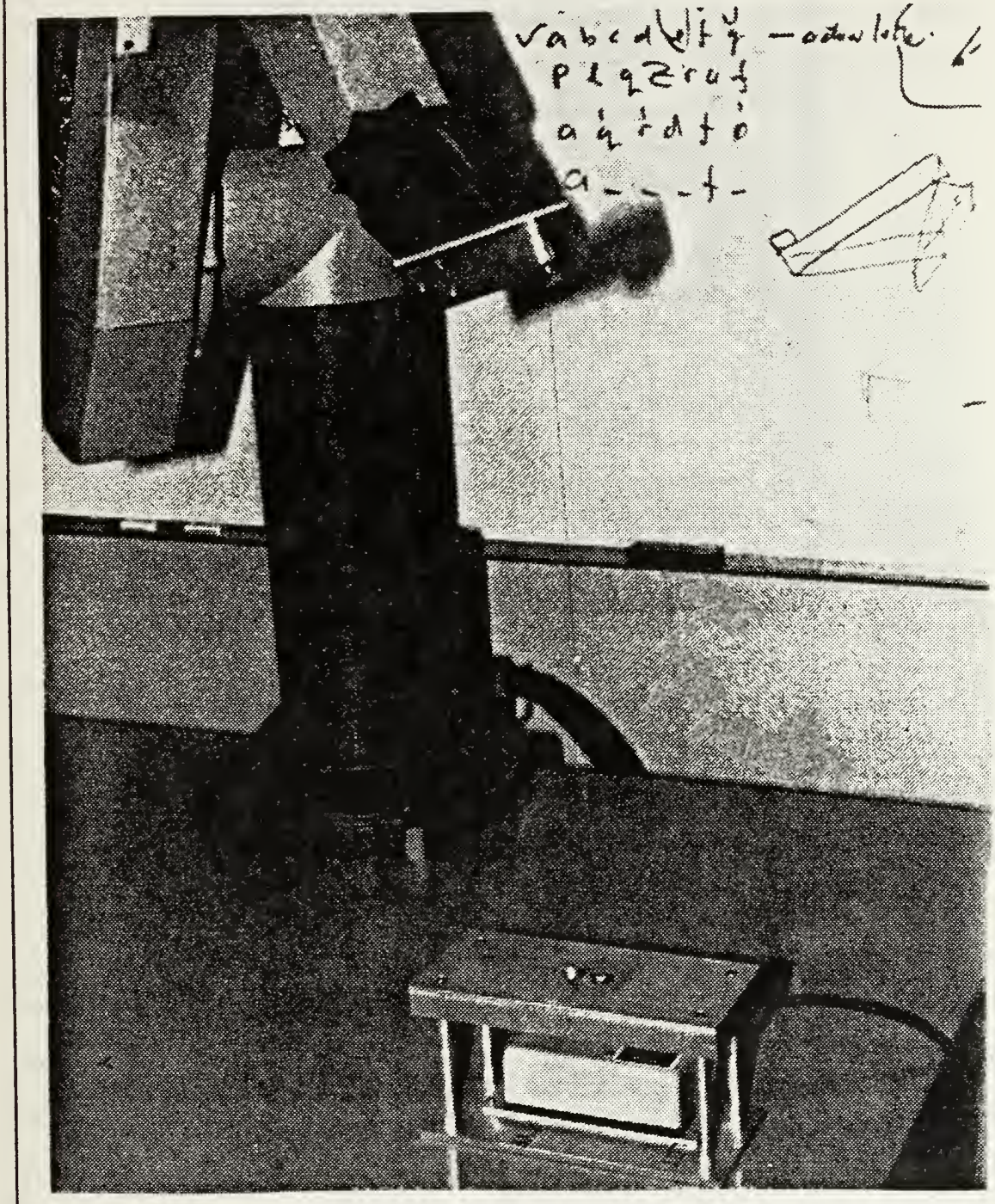

Figure 43. Wire Potentiometer Calibration

Table 8. NOMINAL AND CALIBRATED KINEMATIC PARAMETERS

	Nominal	Calibrated
x_M	-75.0	-75.118
y_M	-711.0	-724.381
z_M	552.0	544.183
a_1	0.0	-0.489
α_1	-90.0	-89.944
$\delta\theta_2$	0.0	-0.523
a_2	431.85	431.958
α_2	0.0	-0.023
β_2	0.0	-0.037
$\delta\theta_3$	0.0	-1.272
d_3	149.09	149.340
a_3	-20.33	-18.735
α_3	90.0	90.125
$\delta\theta_4$	0.0	-0.906
d_4	433.0	432.726
a_4	0.0	0.229
α_4	-90.0	-90.263
$\delta\theta_5$	0.0	1.248
d_5	0.0	-0.532
a_5	0.0	0.136
α_5	90.0	90.254
ϕ_E	-135.0	-131.712
x_E	-76.2	-75.914
z_E	76.2	76.392

The data was then divided into two set's of equal size. The first set of data was used in WID6 for parameter identification. The second set of data was used for verification by computing a forward solution based on the identified parameters and the second sets joint variables. A set of wire lengths was calculated based on the forward kinematic solution results. These wire lengths were then

compared with their corresponding measured values and an rms error of 0.490 mm was calculated.

IV. DISCUSSION OF RESULTS

A. COMPARATIVE ANALYSIS OF EXPERIMENTAL RESULTS

All three calibration techniques resulted in an improvement in the accuracy of the PUMA 560. The resulting accuracy for each method is listed in Table 9.

Table 9. CALIBRATION ACCURACY

COORDINATE MEASURING MACHINE	MODIFIED LINEAR SLIDE	WIRE POTENTIOMETER
0.3 mm	0.744 mm	0.490 mm

Only the 18 parameters of the PUMA are common to each method's closed chain kinematic model. The calibrated values for each method along with the nominal parameters are listed in Table 10 where the length and angular parameters have units of millimeters and degrees respectively. It should be noted that since the actual values are unknown, little can be gained by direct comparison of parameter values. Additionally, repairs were affected to the manipulator between the CMM experiment and the other two methods. The abbreviations CMM, MLS and WP are used in the column headings for the Coordinate Measuring Machine, Modified Linear Slide and Wire Potentiometer methods respectively.

TABLE 10. CALIBRATED PARAMETERS

	NOMINAL	CMM	MLS	WP
a_4	0.00	-0.049	-0.096	-0.489
d_4	-90.00	-89.991	-89.823	-89.944
$\delta\theta_4$	0.00	-0.489	-0.340	-0.523
a_2	431.85	432.122	431.123	431.958
α_2	0.00	-0.030	0.580	-0.023
β_2	0.00	-0.015	0.485	-0.037
$\delta\theta_3$	0.00	-1.207	-0.993	-1.272
a_3	149.09	149.146	146.028	149.340
a_3	-20.33	-19.227	-20.255	-18.735
a_3	90.00	90.051	90.415	90.125
$\delta\theta_4$	0.00	-0.914	-1.089	-0.906
d_4	433.00	432.889	434.095	432.726
a_4	0.00	0.004	0.074	0.229
a_3	-90.00	-89.991	-90.244	-90.263
$\delta\theta_5$	0.00	2.236	1.293	1.248
d_5	0.00	-0.663	-0.993	-0.532
d_5	0.00	-0.026	-0.175	0.136
α_5	90.00	89.934	89.905	90.254

Since all three methods resulted in an accuracy approaching or equalling the repeatability of the manipulator, other factors take on added significance when making comparisons. Some of these factors are somewhat qualitative and they are discussed in the following paragraphs.

Although the CMM base was used for the Modified Linear Slide method, it was used to simulate a device which is assumed could be manufactured with similar measurement

accuracy and characteristics but at a lower cost. With this in mind, the CMM is clearly the most expensive method of the three. The Wire Potentiometer Method is certainly the least expensive.

The compact size of the Wire Potentiometer fixtures make them very portable and durable and hence, well suited to an industrial enviroment. The CMM on the other hand, is clearly better suited for laboratory applications. The Linear Slide, as it did in terms of cost, would seem to fall somewhere between the other two methods. Although a rugged device could be designed, the need for a stiff slide would result in a loss of portability. A stiff slide is necessary to reduce noise induced by flexure under the weight of the manipulator.

Both the CMM and Wire Potentiometer methods are capable of measurements of the PUMA in all manipulator "arm configurations" without any additional considerations. Conversely, great care must be exercised when switching between arm configurations when calibrating with the Modified Linear Slide Method since the end effector must be detached.

Even with automated data acquisition of PUMA joint variable data and CMM position data, it would still require one operator nearly 10 minutes to measure one pose utilizing the CMM. Two operators are required for the Modified Linear Slide Method due to the need to support the PUMA while in Free Mode. Automated data acquisition would greatly reduce the overall time required for this method. The wire potentiometer

method offers significant advantages in terms of data acquisition as well. Automated data acquisition would significantly reduce the amount of time required to gather data. Additionally, the manipulator could be programmed to move through a series of predetermined poses thus enabling the entire process to be automated. The resulting calibration would require only one operator about ten minutes to both set up and dismantle the system. Furthermore, an extensive data base could be collected in a manner of minutes without operator intervention.

B. GENERAL OBSERVATIONS FROM EXPERIMENTS

The original Linear Slide method seemed to offer a number of advantages over the CMM method. Since the end effector is fixed in orientation, six "knowns" are acquired for each displacement measurement. In contrast, the CMM method requires 12 position measurements to "measure" full pose or acquire 6 knowns. Since each position measurement is made with the same accuracy in both methods, it would seem that the Linear Slide method offers a significant reduction in measurement noise. However, the resulting calibration accuracy using the Linear Slide was three times less than achieved with the CMM method [Ref. 13]. This "loss" of accuracy was attributed to additional noise induced by internal loading effects, slide flexure under the manipulator's weight and insufficient joint variable excitation.

Two methods were considered for improving the joint excitation. Appendix G describes a method for determining an optimum position and orientation of the slide. Optimum in this case is defined as a slide position and orientation which results in maximum excursion of all six joint variables as the end effector travels along the slide length. The Modified Linear Slide method offered a significant improvement in joint variable excitation although this method results in a reduction by one half in the number of knowns for each displacement measurement. Although this modification resulted in improvement in the calibration accuracy, the accuracy was still less than that achieved with the CMM method for comparable sized data sets. This reduction in accuracy may be a consequence of mechanical noise as suggested previously. However, simulation studies by Pathre and Driels [Ref. 14] suggest that trajectory or pathlike motion of the end effector may result in a less accurate calibration than if random motion of the end effector is utilized. This is an issue for further research.

The Wire Potentiometer method would seem to support the benefits of both large joint excitation and "random" pose measurements. Despite its noisy characteristics and comparatively lower accuracy, a more accurate calibration was achieved. This suggests that a tradeoff may exist when considering mechanical constraining type measurement systems. In general, additional end effector constraints increase the

number of knowns for each manipulator pose as well as a reduction in noise. However, the additional constraints frequently result in a loss of some "dimension" of the problem and a subsequent loss in calibration accuracy.

C. OBSERVATIONS REGARDING MEASUREMENT SYSTEMS WITHIN CLOSED CHAIN KINEMATIC MODELS

1. INTRODUCTION

As stated earlier, the number of identifiable parameters, N, in a "complete" manipulator kinematic model is given by Equation 31 which is repeated here

$$N = 2P + 4R + 6 \tag{51}$$

where P is the number of prismatic joints and R is the number of rotary joints. For the PUMA 560, N is equal to 30. This model assumes a reference frame external to the manipulator. The advantage of this model is that it offers a convenient method of referencing other objects and tools within the working volume of the manipulator. However, a closed chain kinematic model incorporating a measurement system or device may not provide sufficient information to fully define a "complete" model. Define is used in the sense that all the parameters of the model are identifiable. In general, a manipulator kinematic model incorporating a measurement system will have no more than N identifiable parameters and clearly a model with greater than N parameters contains parameters unnecessary for "completeness".

The process of determining the number of identifiable parameters in models incorporating a variety of measurement systems all to frequently results in an iterative process. For the PUMA and a particular measuring system, the process started with a 30 parameter model followed by simulation studies which resulted in non-convergence if in fact dependencies existed between parameters. The model was then redefined by eliminating parameters based on the numerical results, "first principles" analysis or intuition. Further simulation studies were then conducted until the correct model was developed. Clearly, a systematic approach, such as Denavit-Hartenburg is to manipulator kinematics, would be advantageous for modelling closed loop kinematic chains incorporating different measurement systems.

The ambiguities which can exist are strictly attributable to the measurement system. If the remainder of the model is properly defined, then much of the difficulty is eliminated. The general process is illustrated in Figure 44. The process consists of properly defining a kinematic model of the manipulator. Proper is used in the sense that the model eliminates possible parameter dependencies in transformations to frames external to the manipulator. From this point forward, the term "manipulator kinematic model" or simply "manipulator kinematics" will refer to such a model. The proper kinematic model, in the same sense as before, of the measurement system must then be defined. The transformations

T_M^0 and T_{n-1}^E which connect the two models are then easily defined.

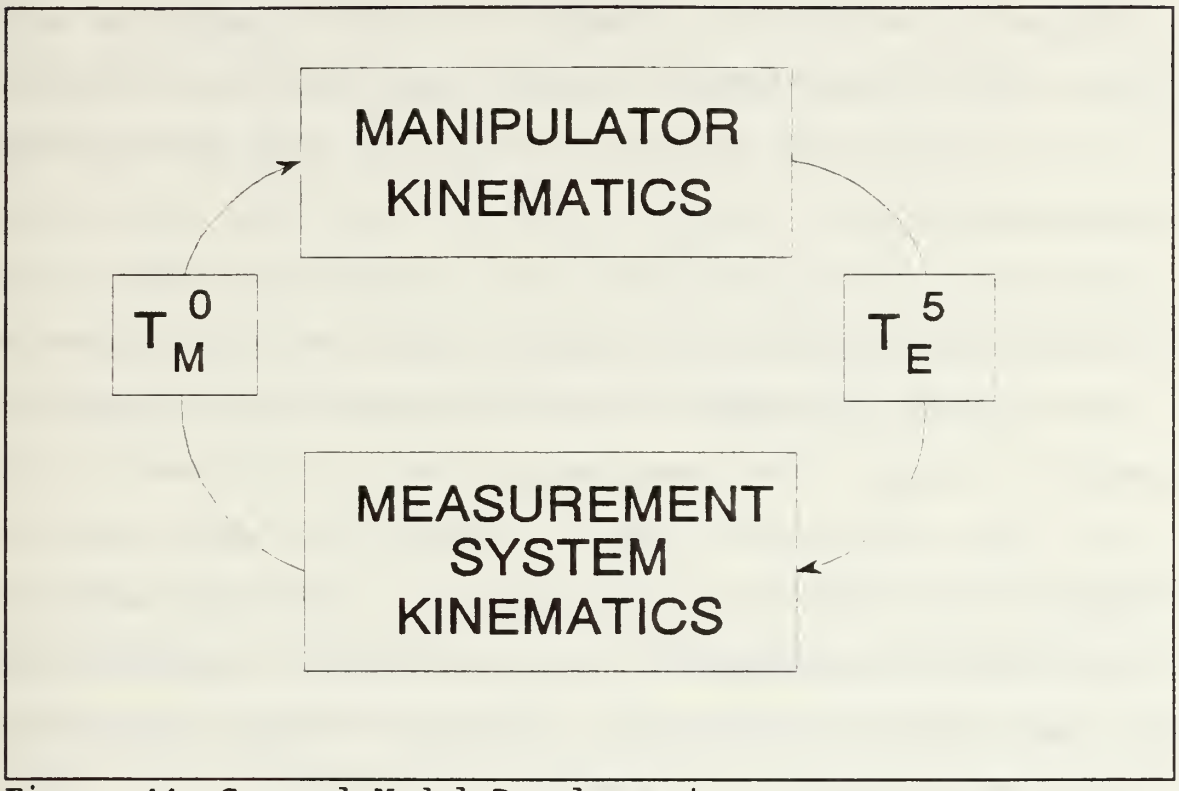

Figure 44. General Model Development

2. Manipulator Kinematic Modelling

For general link to link transformations within a manipulator, the Denavit-Hartenburg or Modified Denavit-Hartenburg method provides a model in which all parameters are identifiable. Care must be exercised though when defining both the base frame of the manipulator and what is defined as the "end" of the manipulator kinematics. For an n link manipulator, the "end" of the manipulator kinematics should be defined by frame n-1. This frame is chosen due to the fact that it is the last frame uniquely defined by the manipulator geometry. Recall though that prismatic joints may lead to

ambiguity in this frame's location as well. A unique manipulator base frame defined by the robots geometry is required as well. For a rotary joint one, in which the joint one axis is not parallel with joint axis two, this is accomplished as described earlier for the PUMA. The resulting transformation to frame one contains only two identifiable parameters. It can be shown that a unique base frame can be defined which results in a similar reduction in the number of identifiable parameters by two in models with parallel or nearly parallel joint axes one and two or a prismatic joint one. This can also be seen by studying the development of Equation 51 [Ref. 15]. The result is that the number of identifiable parameters, K, in a manipulator kinematic model as defined here is given by Equation 52 where P and R are defined as before.

$$K = 2P + 4R - 6 \tag{52}$$

For the six rotary joint PUMA 560, the number of identifiable parameters is 18 as developed earlier and listed in Table 1.

3. Measurement System Modelling

The preceding model development and analysis leads to a lower bound on the number of identifiable parameters in a closed chain kinematics model incorporating a measurement device or system. The last or end frame in a manipulator kinematic model, F^{n-1}, is a unique fixed frame for a given set of joint variables. The base frame is also unique. Therefore, six unique parameters, three rotations and three translations,

are required to transform between the two frames. Any number of parameters less than six would imply dependencies within the manipulator kinematics which are known to be unique. Therefore, the minimum number of parameters, M, in the closed chain model is

$$M = 2P + 4R \tag{53}$$

Summarizing, the number of identifiable parameters, n, in the closed chain kinematic model is bounded by M and N.

$$M \leq n \leq N \tag{54}$$

Applying Equation 54 to the PUMA, the number of identifiable parameters is between 24 and 30 inclusive. In terms of the measurement system kinematic, the foregoing states that the model will consist of at least six and no more than 12 constraints.

4. Linking the Measurement and Manipulator Models

If the number of constraints in the measurement system model is less than 12, then the model will contain geometric quantities such as points or axes which can be thought of as "reduced order frames". As described in the section Other Special Cases, transformations between frames and "reduced order frames" are easily developed and clearly indicate dependant parameters.

5. Case Studies

A number of calibration methods, both performed and proposed, were studied in detail in order to redefine the overall model in the form of Figure 44. Specific emphasis was placed on the measurement system model development with a goal of "standardizing" this process. Each method assumes calibration of a PUMA 560 and thus the manipulator kinematic model consists of the previously defined 18 parameters.

a. The Coordinate Measuring Machine

The Coordinate Measuring Machine method described earlier was studied and the model was then developed in the form of Figure 44. The measurement system, which consist of the CMM and the precalibrated tooling ball end effector, are illustrated in Figure 45 and its kinematic model is briefly restated here. Since the orthogonal axes of the CMM independently define a reference orientation and the position measurements can be made with respect to a zero reference, an independent reference frame is defined. Through a series of position measurements of the calibrated tooling ball end effector, full pose of the end effector is defined as well. Therefore, the measurement system kinematic model consists of two fully defined and independent frames for a total of 12 constraints.

All 12 parameters are identifiable in the transformations linking the manipulator and the measurement system and hence n=30 for the overall model. Table 11 lists

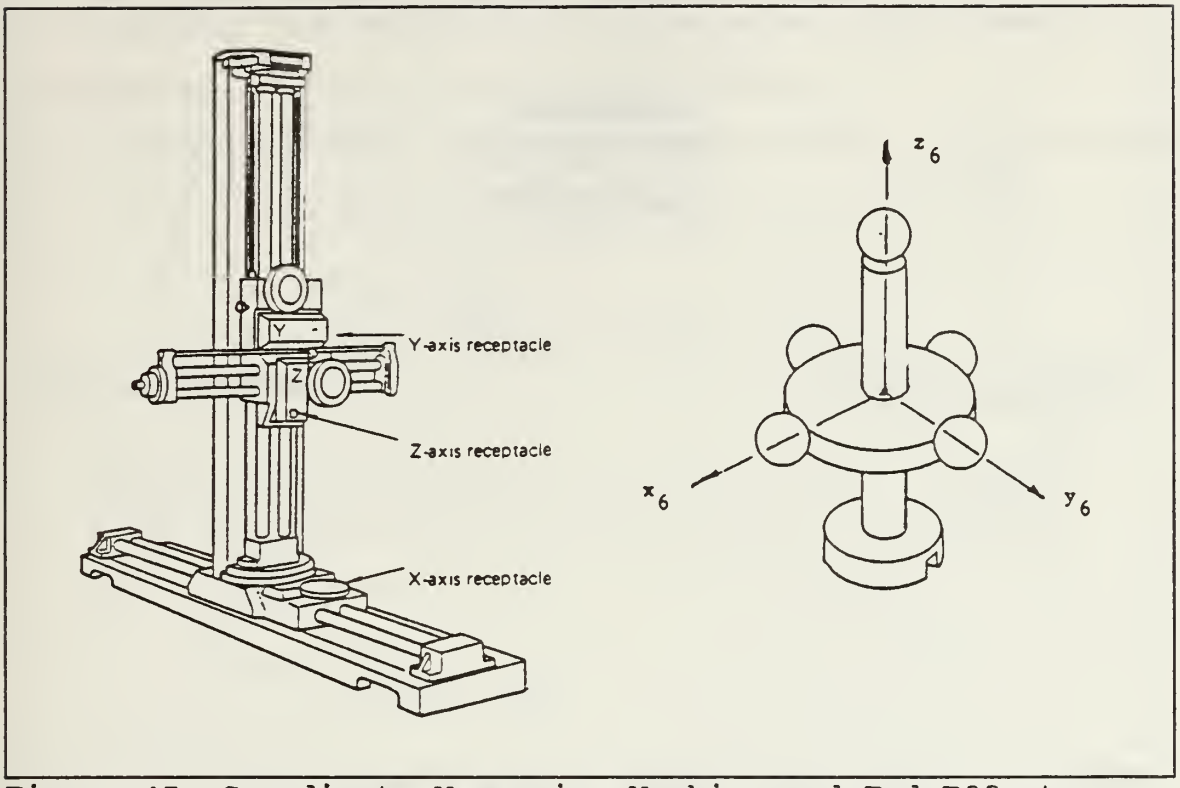

Figure 45. Coordinate Measuring Machine and End Effector

the parameters of T_M^O and T_5^E which are all typed in bold since all 12 are identifiable. Figure 46 illustrates the CMM closed chain kinematic model.

b. The Ball Bar

The ball bar described earlier consist of a rigid bar of fixed length with a ball joint mounted at each end and is illustrated in Figure 47. One ball joint is fixed in the manipulator's workspace and the other is attached to the PUMA's end effector flange. Each ball joint has three constraints, fixed translations, and consequently is capable of defining a point. The three degrees of freedom, three rotations, prevent associating an orientation at either joint.

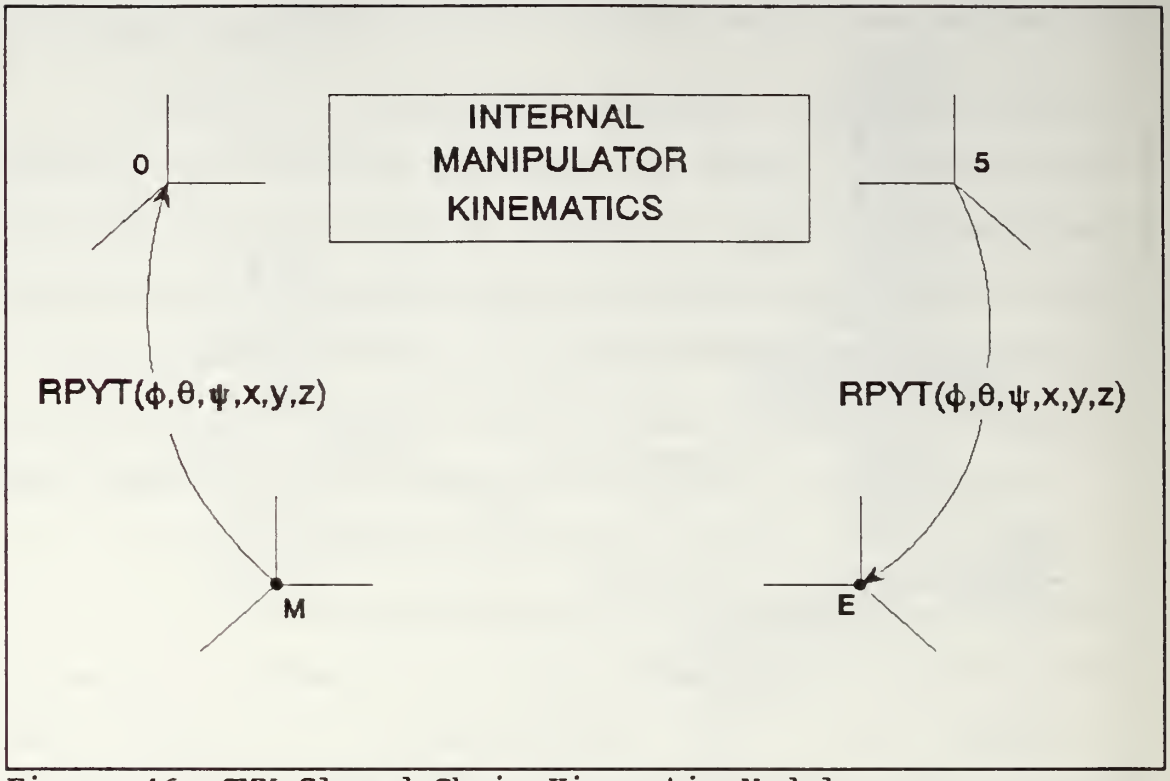

Figure 46. CMM Closed Chain Kinematic Model

Table 11. T_M^O and T_5^E Kinematic Parameter Table

T_M^O	T_5^E
ϕ_M	ϕ_E
θ_M	ψ_E
ψ_M	x_E
x_M	y_E
y_M	z_E
z_M	

Consequently, the measurement system model simply consist of two points.

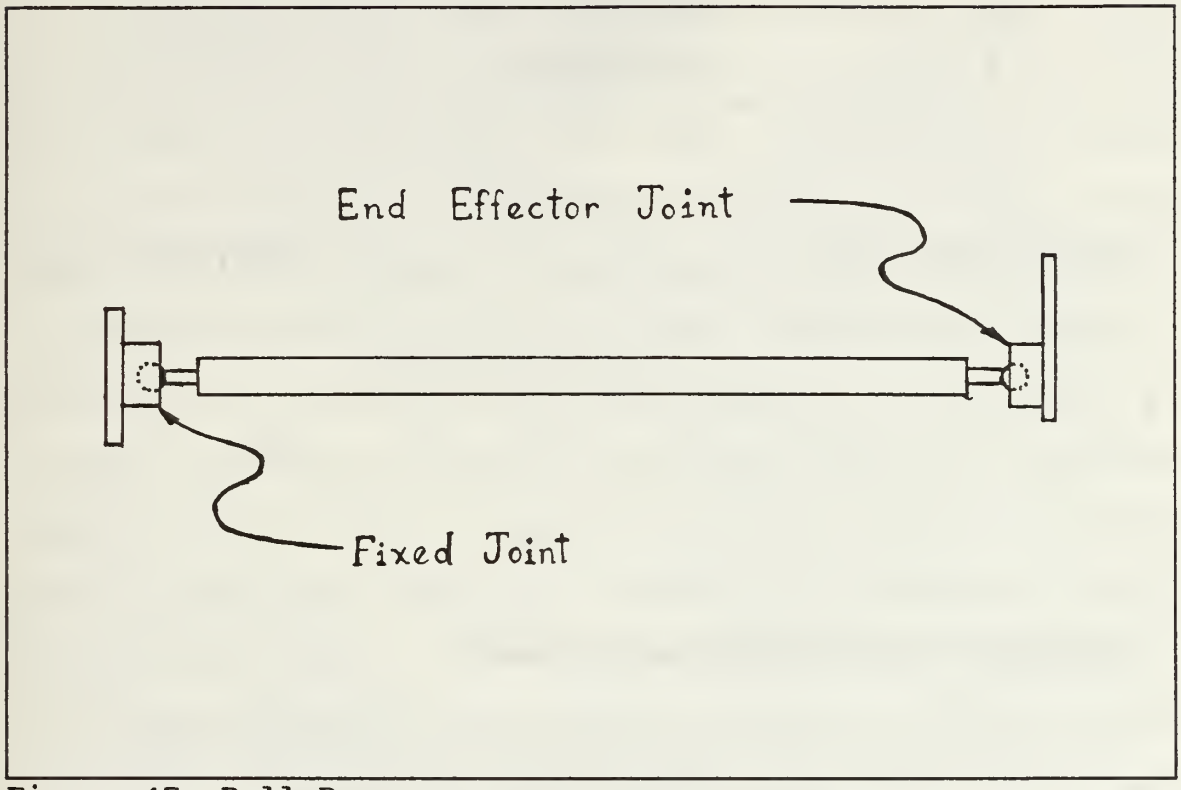

Figure 47. Ball Bar

As noted earlier, three parameters are required to transform from a frame to a point and one of these parameters must be a translation. The overall model must then consist of 24 identifiable parameters. A logical choice of parameters for identification are indicated in bold in Table 12. The unidentifiable parameters of the RPYT transformation are defined to be zero as indicated. The closed chain kinematic model is illustrated in Figure 48.

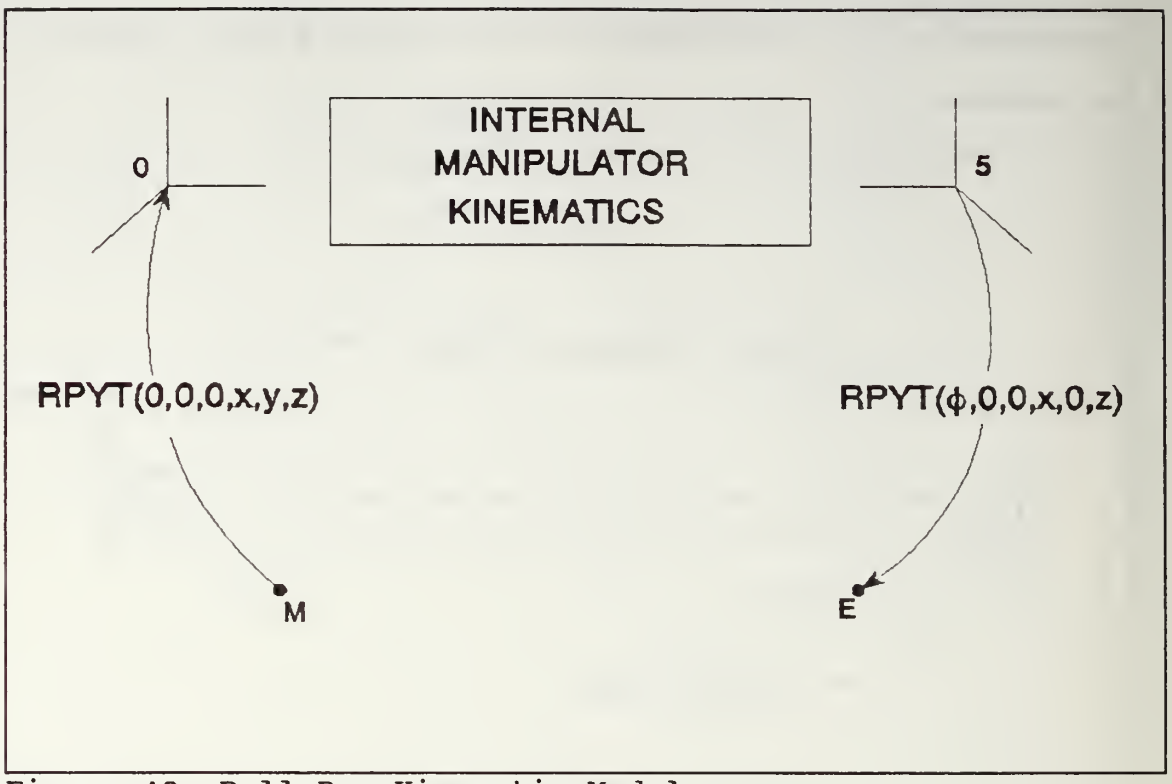

Figure 48. Ball Bar Kinematic Model

Table 12. T_M^O and T_5^E Kinematic Parameter Table

T_M^O	T_5^E
0	ϕ_E
0	0
0	0
x_M	ϕ_E
y_M	0
z_M	z_E

c. The Linear Slide

The linear slide measurement system consists of the lower base assembly of the CMM. The y axis post is removed and the PUMA end effector is bolted in this location. The PUMA and slide are illustrated in Figure 49. The end effector orientation, since it is bolted to the slide, is fixed. Motion along the slide defines a direction. Relative displacement measurements are available from the CMM display unit. However, much like the prismatic joints in manipulator kinematics, no specific point is defined by the geometry. In an analogous fashion, an axis can be specified at the "next" or "last" defined coordinate frame. Two choices are available, either the PUMA base frame or frame 5. Defining the axis at frame 5 more closely resembles the physical system and so this frame is chosen. This fixes F^E coincident with F^5. With an axis now defined, a unique point on this axis can be defined by the common normal between joint axis one and this axis. Another fixed point on the axis is defined as well by the relative displacement measurements. The origin of F^M can be placed at this point noting that only one direction or orientation constraint is specified. Summarizing, the orientation of F^E is specified but its origin is not unique. The origin of F^M is specified as well as one orientation.

To transform from a frame to a point on axis requires three translations (point to point) and two rotations for an axis alignment. Since F^E is coincident with F^5, only

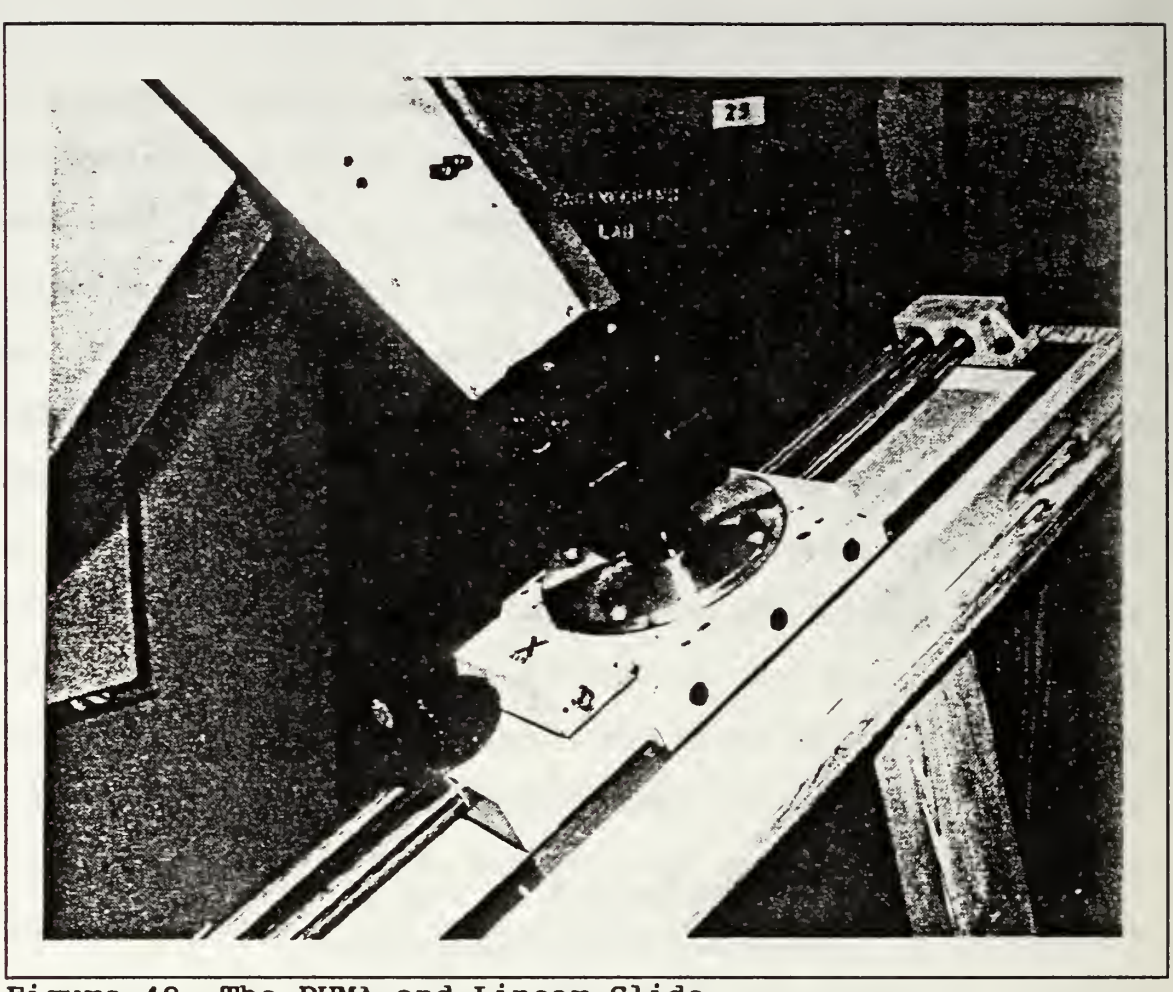

Figure 49. The PUMA and Linear Slide

three rotations are necessary to transform between these two frames. Therefore, eight parameters are necessary for the transformations between the manipulator kinematics and the measurement kinematics which results in a 26 parameter model. Table 13 lists the parameters in these two transformations with the identifiable parameters in bold and the unidentifiable parameters defined to be zero as before. The model is illustrated in Figure 50.

Table 13. T_M^O and T_S^E Kinematic Parameter Table

T_M^O	T_S^E
ϕ_M	ϕ_E
$\dot{\phi}_M$	θ_E
0	ψ_E
x_M	0
y_M	0
z_M	0

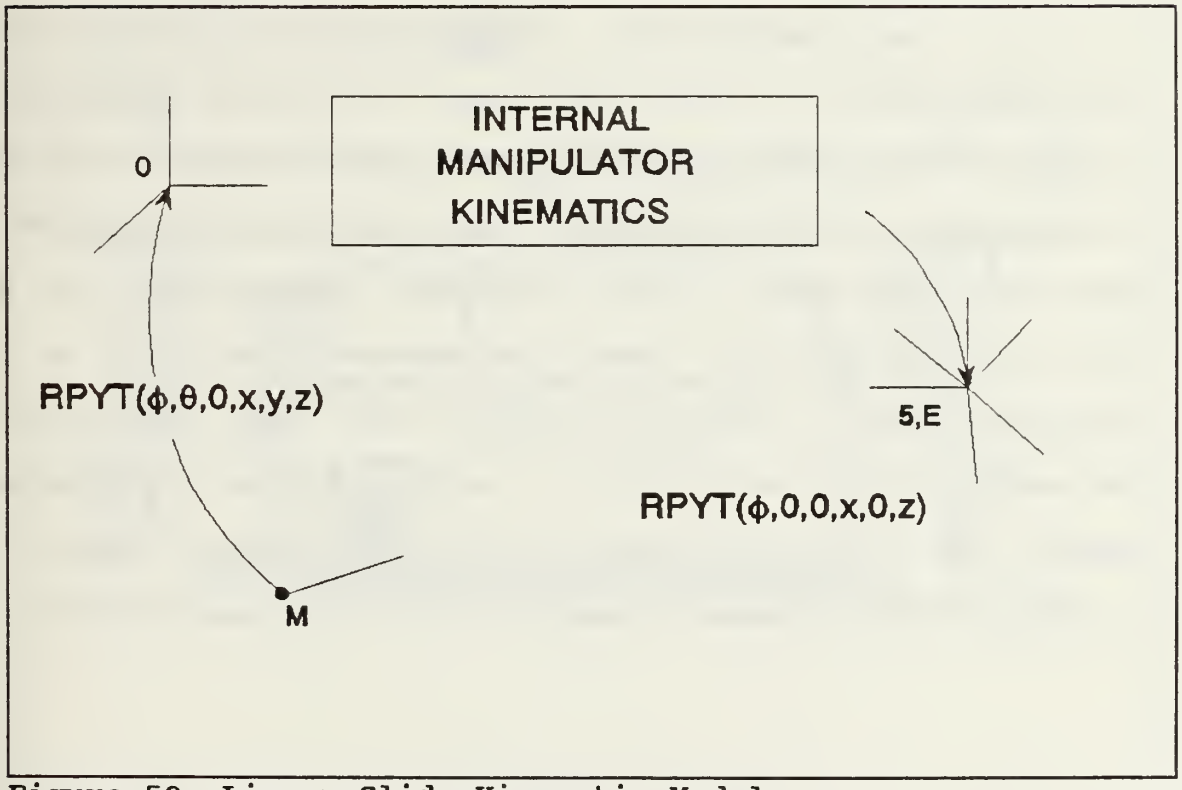

Figure 50. Linear Slide Kinematic Model

d. The Wire Potentiometer

The Wire Potentiometer method was described in detail earlier and is briefly described here. The wire potentiometer provides a resistance which is proportional to the amount of wire extracted from the device. These devices are designed for linear displacement measurements. In order to provide displacement measurements in a volume, the two fixtures illustrated in Figure 51 were designed. The funnel shaped ports prevent wire deformation and each defines an axis. The throat of the funnel defines a fixed point in a similar fashion to a ball joint. Therefore, the measurement system consists of two points and an axis through each.

As noted earlier, five parameters are required to transform between a frame and a point on an axis. Therefore, a model with 28 identifiable parameter results. Note that due to the small size of the funnel ports used in the previously described experiment, both axes were assumed not to be identifiable. In this case, the model reduces to a 24 parameter model. Table 14 lists both the identifiable parameters and unidentifiable as before for the 28 parameter model. Note that other combinations are possible. Figure 52 illustrates the kinematic model.

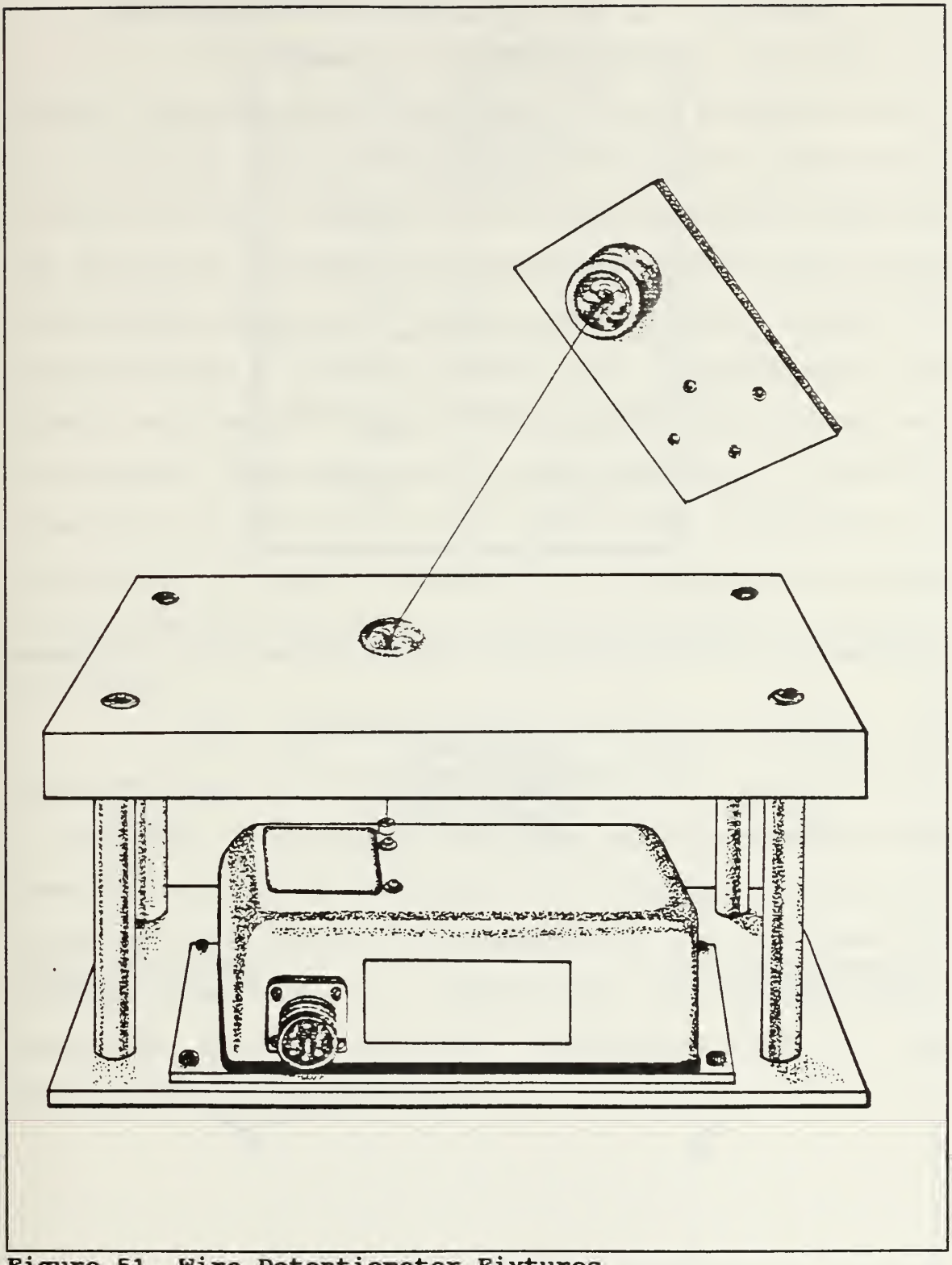

Figure 51. Wire Potentiometer Fixtures

Table 14. T_M^O and T_5^E Kinematic Parameter Table

T_M^O	T_5^E
ϕ_M	ϕ_E
θ_M	θ_E
$\blacklozenge_M$	$\blacklozenge_E$
$\blacklozenge_M$	x_E
$\blacklozenge_M$	y_E
z_M	z_E

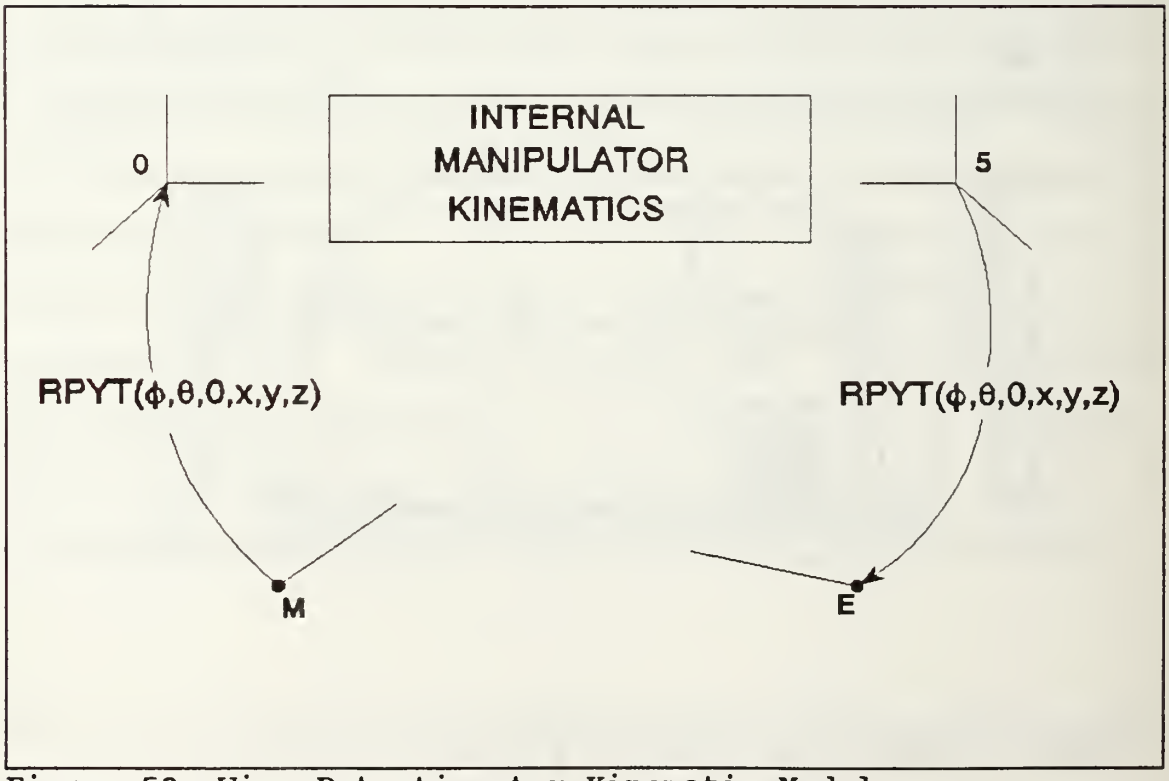

Figure 52. Wire Potentiometer Kinematic Model

e. Single Theodolite

Calibration of a PUMA 560 was performed by Whitney, Lozinski and Rourke using a single theodolite [Ref. 16]. The single theodolite measurement system is composed of a theodolite and a target fixed on the manipulators endpoint. The theodolite is capable of accurately measuring an azimuth and elevation angle of an object along its line of sight. The target defines a point in space. The intersection of the azimuth and elevation angles defines a point and since these are measured with respect to a zero reference, a specific orientation is defined at this point. This relationship is illustrated in Figure 53. Summarizing, the measurement system kinematic model consists of a point at F^E and a fully defined frame at F^M.

Six parameters are obviously necessary to transform from F^M to F^O. The transformation T_5^E requires three of parameters and as usual, one must be a translation. With these 9 identifiable parameters, the resulting closed chain kinematic model consists of 27 parameters. Table 15 lists one possible combination of identifiable and unidentifiable parameters. Figure 54 illustrates the kinematic model for the single theodolite closed chain.

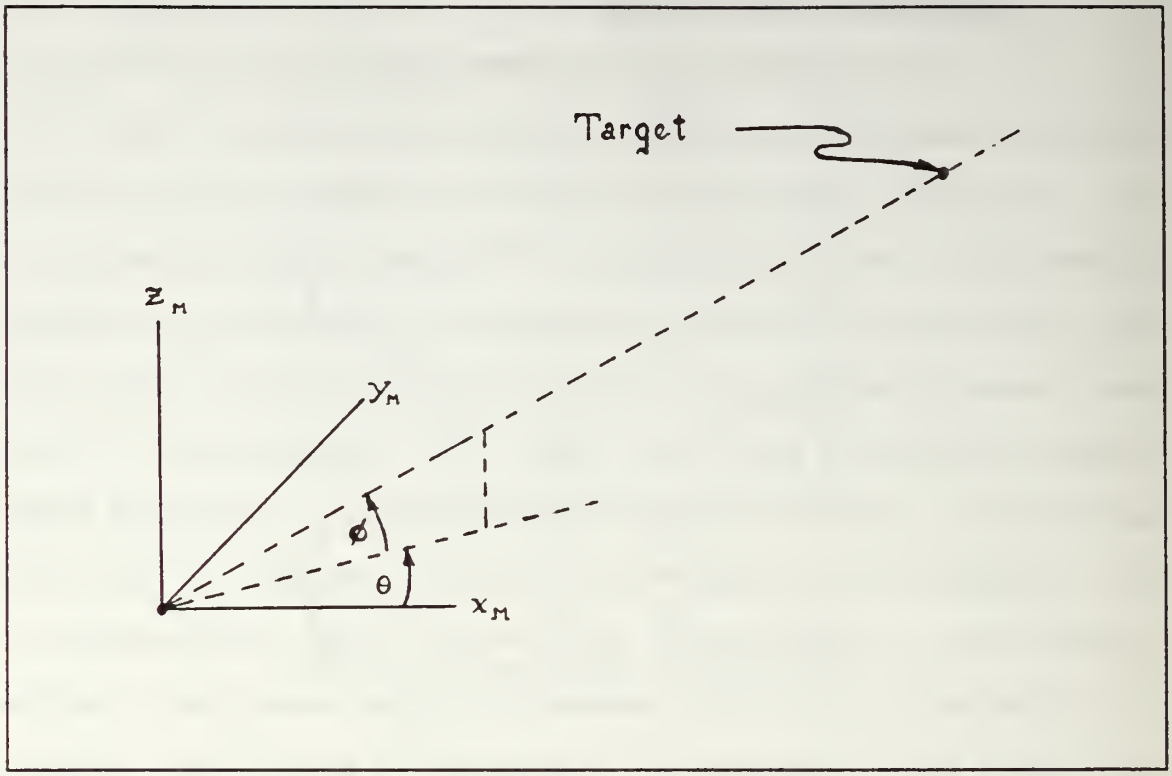

Figure 53. Theodolite Measurement System

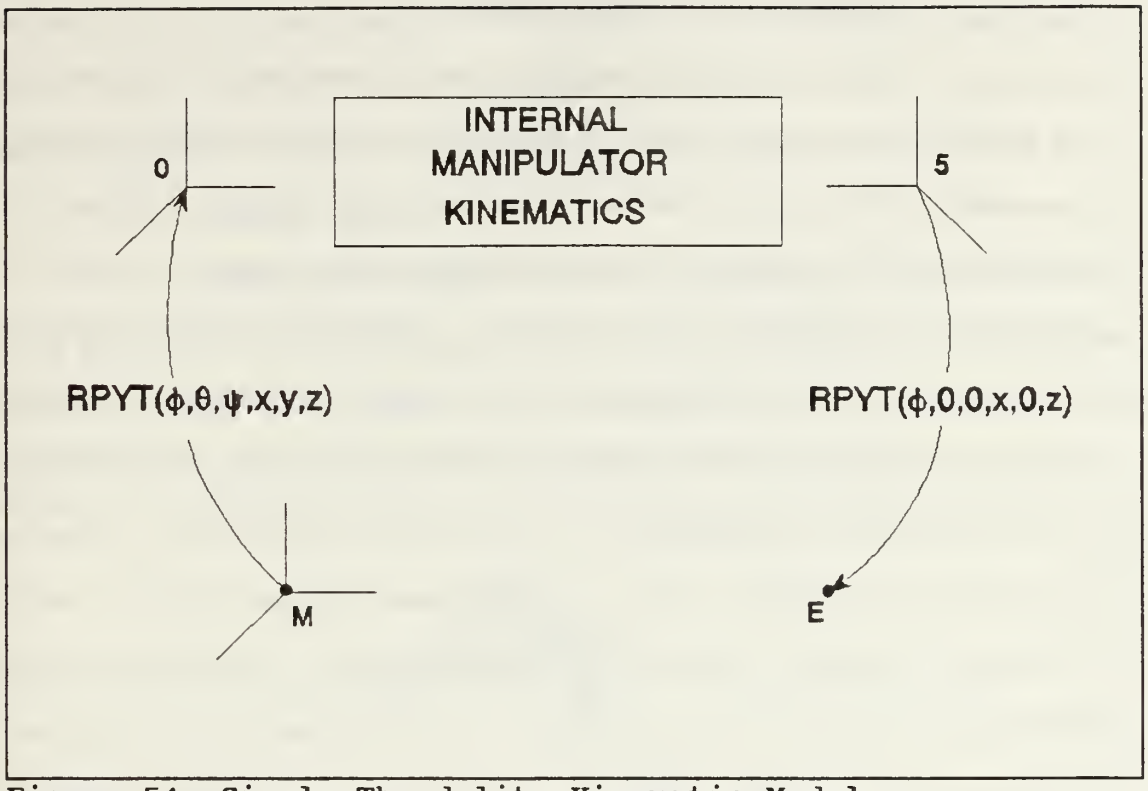

Figure 54. Single Theodolite Kinematic Model

Table 15. T_M^O and T_5^E Kinematic Parameter Table

T_M^O	T_5^E
ϕ_M	ϕ_E
θ_M	0
ψ_M	0
$\spadesuit_M$	$\spadesuit_E$
$\bullet_M$	0
z_M	z_E

f. Three Wire Potentiometer

The three wire potentiometer method utilizes three wire potentiometers placed in a known or calibrated triangular arrangement. The end effector fixture consists of a triangular shaped plate with three funnel shaped ports again placed in a calibrated triangular arrangement. A sketch of the system is shown in Figure 55. The known triangular arrangement defines a frame at both the measurement system base and end effector.

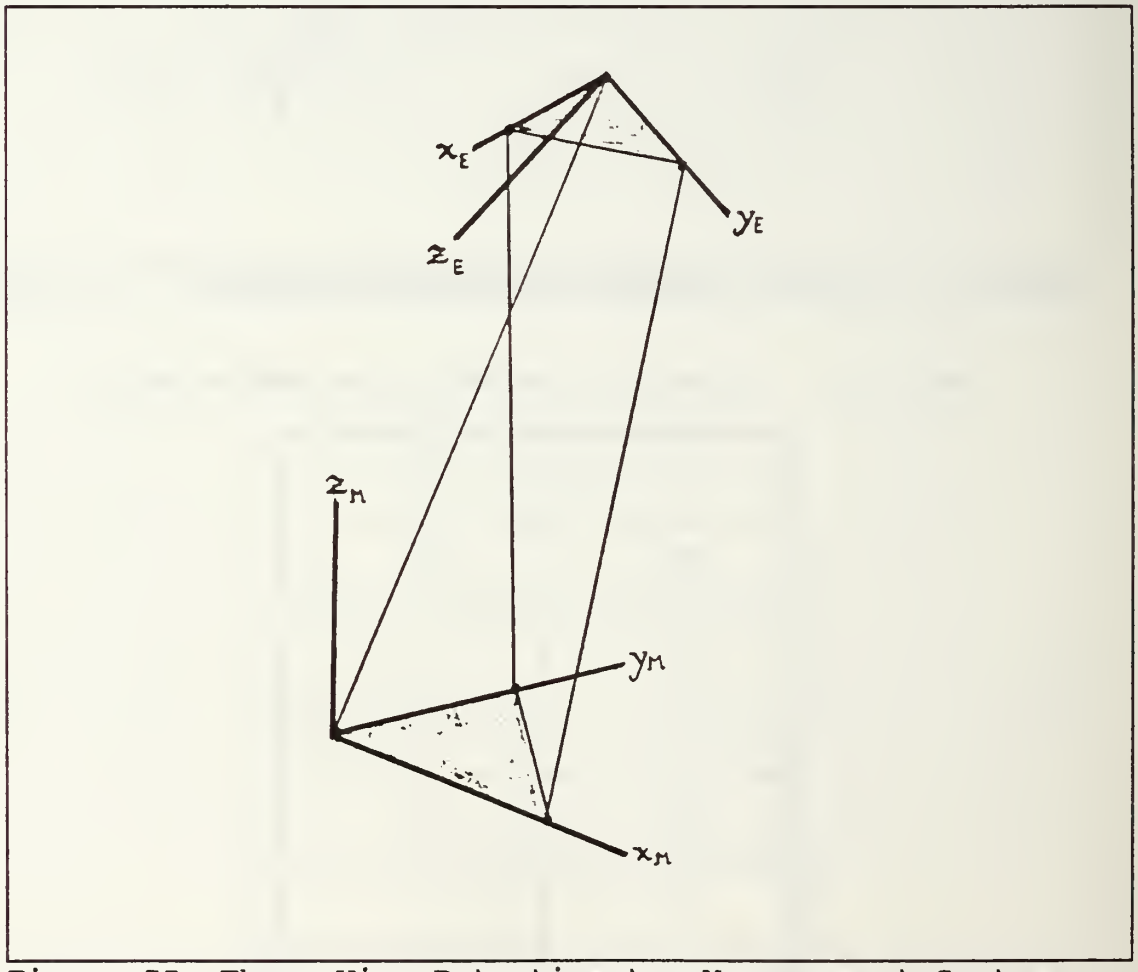

Figure 55. Three Wire Potentiometer Measurement System

With frames defined at both F^M and F^E, the resulting closed chain kinematic model contains all 30

parameters of the "complete" model. The kinematic parameter table for T_M^0 and T_5^E is the same as Table 11 and the closed chain kinematic model is the same as Figure 46.

g. Planar Motion

This measurement system consist of a smooth flat plate placed in the manipulators workspace and an end effector with a single precision tooling ball. The tooling ball is placed in contact with the plate in a variety of joint variable configurations and at arbitrary locations on the plate. In this measurement system, the tooling ball defines a point and this is the makeup of F^E. The "planar" motion of this point defines a plane. A unique point can be defined at the intersection of the plate "plane" and joint axis one. A unique direction perpendicular to the plate is specified at this point but no additional orientation is specified. Consequently, F^M is composed of a point on an axis. Note that F^M lies on joint axis one.

Two rotations and one translation are required in T_M^0. The rotations are necessary for axis alignment and then the translation along joint axis one places F^M at F^0. Three parameters can be identified in T_5^E, a frame to point transformation, and one of these must be a translation as usual. The resulting closed chain kinematic model then consists of 24 identifiable parameters. Table 16 indicates one possible combination of identifiable parameters. Figure 56 depicts the closed chain kinematic model. If the tooling ball

end effector of CMM method were used instead of a single tooling ball, then a frame is defined at F^E and a 27 parameter model results.

Table 16. T_M^O and T_S^E Kinematic Parameter Table

T_M^O	T_S^E
ϕ_M	ϕ_E
ϕ_M	θ_E
0	0
0	0
0	0
z_M	z_E

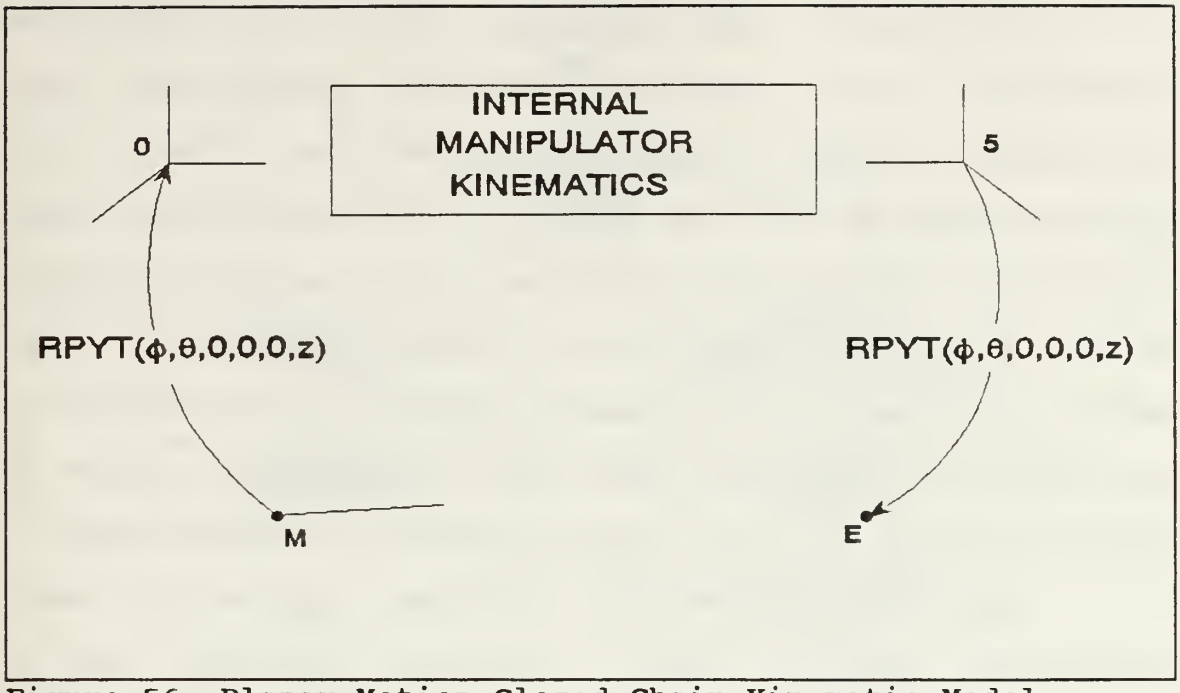

Figure 56. Planar Motion Closed Chain Kinematic Model

6. Discussion of Case Studies

There are, in general, two broad categories of measurement systems used in manipulator calibration:

- Those which mechanically constrain the manipulator
- Those which do not mechanically constrain the manipulator

The non-constraining type systems are typically easier to model and these will be discussed first.

One reason that non-constraining type measurement systems are usually easier to model is that a measurement system reference frame is typically better defined. The second reason is that the measured quantities are often the more familiar cartesian or spherical coordinates of a point or points on the end effector.

Conversely, the location and any associated orientation of a reference frame for constraining type measurement systems is frequently not obvious. In fact, it is often not defined until the position of the end effector frame is defined. In this case, this *frame* is not really a reference at all, but it is convenient for symmetry to refer to it as such. Additionally, the measured quantities may consist of some known spatial relationship such as a path or surface as opposed to more familiar lengths and angular displacement.

Although no foolproof cookbook approach to model development seems to exist, a set of guidelines can be established. A generalized process for model development is illustrated in the flowchart in Figure 57. The first step is to determine which of the two broad categories the measurement system belongs. Even this step is not always trivial since a measurement system such as the wire potentiometer could be arguably placed in either category.

As stated earlier, most non-constraining measurement systems have relatively well defined reference frames. However, it is noted that it is not a trivial task to define why, for example, the CMM, which is a system of three prismatic joints, establishes a unique reference frame even though it seems quite obvious at first glance.

Once a reference frame is established, the end effector *frame* is analyzed. Most non-constraining type measurement systems measure points on the end effector in

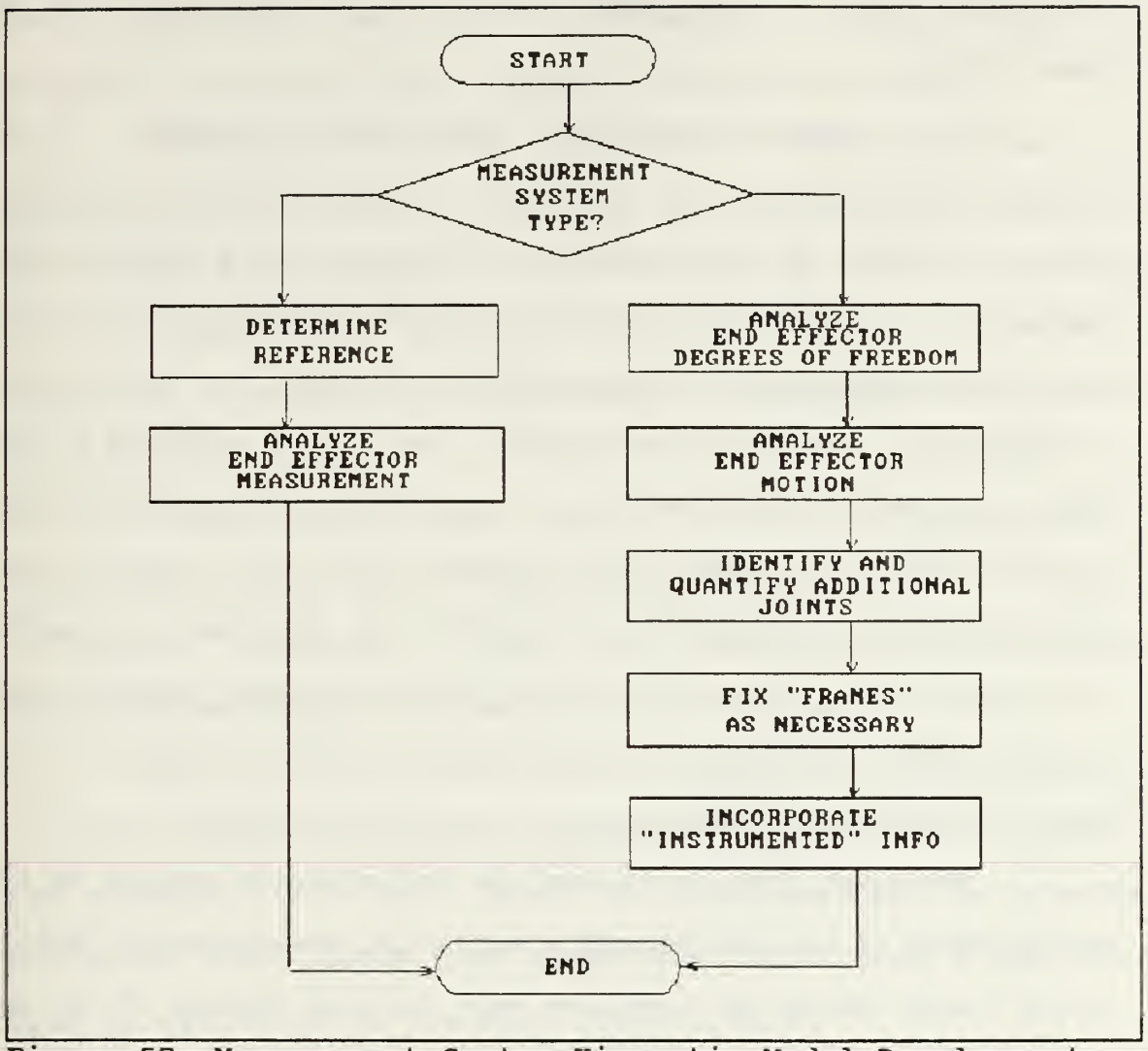

Figure 57. Measurement System Kinematic Model Development

either spherical or cartesian coordinates or can be modelled in this fashion. It is not necessary for the point to be fully defined to be modelled as such as in the case of the theodolite. In this example, the azimuth and elevation angles are known but the distance between the reference frame and the point is unknown. If additional points or other geometric factors are measured for each set of joint variables, then

full pose or at least some aspects of orientation can be identified.

For constraining type measurement systems, it is usually advantageous to analyze the end effector fixture first. Analysis of the degrees of freedom of the end effector fixture at an arbitrary point will begin to define the make-up of F^E. For example, the end effector fixture for the Linear Slide method, which is essentially the x axis carriage of the CMM, is fixed in orientation at any position along the slide. In the Modified Linear Slide method, the ball joint allows three rotational degrees of freedom. The degrees of freedom for mechanical joints are easily identified and provide much of the needed information. The result of this analysis may or may not define a fixed point or location in space.

The next step is to analyze end effector motion as the manipulator is varied through a series of measurements. Motion along known paths or surfaces may further define F^E or may begin to define the location of F^M. For example, a unique reference point was established by the plane defined by end effector motion in the Planar Motion calibration example.

At this point in the process, additional joints or features of the measurement system should be identified. Additional joints will likely establish the reference *frame* origin. For example, the fixed ball joint in the Ball Bar method or the funnel throat in the Wire Potentiometer method define the origin of F^M. However, this point would already be

established in the Ball Bar method by considering the spherical surface traced out by end effector motion.

Specific or unique locations of each *frame* may be defined when this step is reached. However, as in the case of the Linear Slide method, no unique point in space is defined by the measurement system considered alone since only a direction is defined. As illustrated in the Linear Slide model development, either F^M or F^E must be defined coincident with either the manipulator base frame or the last defined frame of the manipulator kinematic model. When this assignment is made, additional parameters of the other measurement system *frame* may be fixed based on geometric relationships.

The final step involves incorporating "instrumented" information provided by the measurement system into the model. For example, if a ball joint is "instrumented" to provide elevation angle above some prescribed zero, then a specific axis is specified at the ball joint origin.

The large scope of measurement systems and devices capable of being used in manipulator calibration defies a more rigorous algorithm for model development. However, the preceding approach does provide a sound approach which will result in a properly defined model.

V. CONCLUSIONS

A. EXPERIMENTAL RESULTS

- A PUMA 560 manipulator can be calibrated using either one of the three methods with a resulting accuracy which approaches or equals the repeatability of 0.3 mm.

- The Coordinate Measuring Machine calibration method will identify the "complete" 30 parameter kinematic model with a resulting accuracy of 0.3 mm.

- The Modified Linear Slide method will identify 26 parameters with a resulting accuracy of 0.74 mm.

- The Wire Potentiometer will identify 24 parameters with a resulting accuracy of 0.49 mm.

- The Modified Linear Slide method did result in a more accurate calibration, 0.74 mm, than the Linear Slide method, 0.9 mm, which emphasizes the need for large joint variable excitation.

- The lower calibration accuracy achieved with the Linear Slide Method as compared to the less accurate Wire Potentiometer supports the theory that "random" poses are more desirable than poses restricted to paths or trajectories regardless of joint excitation.

- The Wire Potentiometer method provides an accurate, inexpensive, portable calibration device which is easily automated and capable of rapid calibration of a large class of manipulators.

B. MEASUREMENT SYSTEMS WITHIN CLOSED CHAIN KINEMATIC MODELS

- A manipulator kinematic model can be developed which will exhibit no parameter dependancies within transformations to external reference frames

- The number of identifiable parameters of a typical serial link manipulator in a closed chain kinematic model is given by Equation 52 and for the PUMA 560 the number of parameters is 18.

- The process of developing closed chain kinematic models with embedded measurement systems for parameter identification can be divided into three separate tasks:

Develop the "manipulator kinematic model"; Develop the "measurement system kinematic model"; Link the two models using only the required parameters of a RPYT transformation.

- No comprehensive algorithm for measurement system kinematic model development exists due to the wide variety of measurement systems which can be employed. However, some useful guidelines may be employed.

APPENDIX A

```
      PROGRAM BLINSC


C  This program generates sets of joint angles for the Puma manipulator
C  arm.  It assumes that the tool frame of the manipulator is
C  constrained to move in the positive x direction only.  The
C  tool is constrained by a ball joint mounted to a sliding linear scale.
C  The values along the x direction are generated by a random number
C  generator.

      INTEGER LDFJAC, M, N, OBS, NOBS
      PARAMETER (LDFJAC=3, M=LDFJAC, N=6)

      REAL*8 fi0, th0, si0, px0, py0, pz0
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6

      REAL*8 RN1,RN2,RN3,RN4,RN5,RN6
      REAL*8 RN7,RN8,RN9,RN10,RN11,RN12
      REAL*8 RN13,RN14,RN15,RN16,RN17,RN18

      INTEGER INFER,IER,IOPT,NSIG,MAXFN
      REAL*8 FJAC(LDFJAC,N), XJTJ((N+1)*N/2), XJAC(LDFJAC,N)
      REAL*8 PARM(4), F(LDFJAC), WORK((5*N)+(2*M)+((N+1)*N/2))
      REAL*8 X(N)
      REAL*8 MAGNX,MAGN1

      EXTERNAL PUMA_ARM

      INTEGER I, J, K, NOU
      REAL*8 TDES(3), T(4,4), SCALE, DANGLE, DLENTH, NUM

      COMMON /PDATA/ TDES, DANGLE, DLENTH, T
      COMMON /KIN/ FI0,TH0,SI0,PX0,PY0,PZ0
     &             DT1,DT2,DT3,DT4,DT5,
     &             AL1,AL2,AL3,AL4,AL5,
     &             AA1,AA2,AA3,AA4,AA5,
     &             DD1,DD2,DD3,DD4,DD5,
     &             BL1,BL2,BL3,BL4,BL5,
     &             DF6,TH6,SI6,PX6,PY6,PZ6
```

```
C Initialize data variables

      OBS=0

C Open data files for input and output

      OPEN (10, NAME='PUMA-POS.DAT', STATUS='NEW')
      OPEN (9, NAME='INPUT.DAT', STATUS='OLD')

C Read input kinematic data

      READ (9,*)
      READ (9,*) FI0,TH0,SI0,PX0,PY0,PZ0
      READ (9,*) DT1,DD1,AA1,AL1,BL1
      READ (9,*) DT2,DD2,AA2,AL2,BL2
      READ (9,*) DT3,DD3,AA3,AL3,BL3
      READ (9,*) DT4,DD4,AA4,AL4,BL4
      READ (9,*) DT5,DD5,AA5,AL4,BL5
      READ (9,*)
      READ (9,*) DF6,TH6,SI6,PX6,PY6,PZ6
      READ (9,*)
      READ (9,*) NOBS,NOU,DANGLE,DLENTH,MAGNX,MAGNL

      CLOSE (9)

C Adjust nominal values

      FI0=FI0+DANGLE
      TH0=TH0+DANGLE
      SI0=0.0
      PX0=PX0+DLENTH
      PY0=PY0+DLENTH
      PZ0=PZ0+DLENTH

      DT1=0.0
      DT2=DT2+DANGLE
      DT3=DT3+DANGLE
      DT4=DT4+DANGLE
      DT5=DT5+DANGLE

      AL1=AL1+DANGLE
      AL2=AL2+DANGLE
      AL3=AL3+DANGLE
      AL4=AL4+DANGLE
      AL5=AL5+DANGLE

      AA1=AA1+DLENTH
      AA2=AA2+DLENTH
      AA3=AA3+DLENTH
      AA4=AA4+DLENTH
      AA5=AA5+DLENTH
```

```
      DD1=0.0
      DD2=0.0
      DD3=DD3+DLENTH
      DD4=DD4+DLENTH
      DD5=DD5+DLENTH

      BL1=BL1
      BL2=BL2+DANGLE
      BL3=BL3
      BL4=BL4
      BL5=BL5

      DF6=DF6+DANGLE
      TH6=TH6
      SI6=SI6
      PX6=PX6+DLENTH
      PY6=PY6
      PZ6=PZ6+DLENTH

C Get random number seed

      WRITE (6,*) 'Type in a 6-digit random number seed'
      READ (5,*) ISEED

C Start of main loop

 1010 OBS=OBS+1

C Set initial values of joint variables

        X(1)=70.0
        X(2)=0.0
        X(3)=90.0
        X(4)=0.0
        X(5)=50.0
        X(6)=90.0

C Get random slide lengths

 1000   CALL RANDOM (ISEED,NUM)
        NUM=NUM*940.0

C Establish desired tool position

        TDES(1)= NUM
        TDES(2)= 0.0
         TDES(3)= 0.0

C Call IMSL ZXSSQ for inverse kinematic solution

        NSIG=4
```

```
      EPS=0.0
      DELTA=0.0
      MAXFN=500
      IOPT=1
      IXJAC=LDFJAC

      CALL ZXSSQ(PUMA_ARM,M,N,NSIG,EPS,DELTA,MAXFN,IOPT,PARM,X,
     &           SSQ,F,XJAC,IXJAC,XJTJ,WORK,INFER,IER)

C Check for singularities

      IF (SSQ .GT. 0.00001) GOTO 1000

C Print results to 2 decimal places

      WRITE (6,*) OBS,SSQ

C Generate the random noise

      CALL RANDOM (ISEED,RN1)
       CALL RANDOM (ISEED,RN2)
      CALL RANDOM (ISEED,RN3)
      CALL RANDOM (ISEED,RN4)
      CALL RANDOM (ISEED,RN5)
      CALL RANDOM (ISEED,RN6)
      CALL RANDOM (ISEED,RN7)
      CALL RANDOM (ISEED,RN8)
      CALL RANDOM (ISEED,RN9)

      RN1 =  MAGNX * (2.0 * RN1 - 1.0)
      RN2 =  MAGNX * (2.0 * RN2 - 1.0)
      RN3 =  MAGNX * (2.0 * RN3 - 1.0)

      RN4 =  MAGN1 * (2.0 * RN4 - 1.0)
      RN5 =  MAGN1 * (2.0 * RN5 - 1.0)
      RN6 =  MAGN1 * (2.0 * RN6 - 1.0)
      RN7 =  MAGN1 * (2.0 * RN7 - 1.0)
      RN8 =  MAGN1 * (2.0 * RN8 - 1.0)
      RN9 =  MAGN1 * (2.0 * RN9 - 1.0)

C Inject random noise

      X(1) = X(1) + RN4
      X(2) = X(2) + RN5
      X(3) = X(3) + RN6
      X(4) = X(4) + RN7
      X(5) = X(5) + RN8
      X(6) = X(6) + RN9

       TDES(1)=TDES(1)+RN1
       TDES(2)=TDES(2)+RN2
```

```
        TDES(3)=TDES(3)+RN3

        WRITE (10,*) X(1),X(2),X(3),X(4),X(5),X(6)
        WRITE (10,*) TDES(1),TDES(2),TDES(3)
        WRITE (10,*)

C Continue for other slide positions

        IF (OBS .LT. NOBS) GOTO 1010

        CLOSE (10)

      END

C ***************************************************************

      SUBROUTINE PUMA_ARM (X,M,N,F)

C This subroutine calculates the non-linear function for the use of
C the IMSL routine ZXSSQ. It is the inverse kinematic solution for
C the PUMA manipulator.

      INTEGER M, N
      REAL*8 X(N), F(M)

      INTEGER II, JJ
      REAL*8 FI0, TH0, SI0, PX0, PY0, PZ0
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6

      REAL*8 TH1, TH2, TH3, TH4, TH5
      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4), T4(4,4)
      REAL*8 T5(4,4), T6(4,4), trpy(4,4), txyz(4,4)
      REAL*8 TIMAT(4,4), T(4,4), td(4,4)

      INTEGER I, J, K
      REAL*8 TDES(3), DANGLE, DLENTH, SCALE

      COMMON /PDATA/ TDES, DANGLE, DLENTH, T
      COMMON /KIN/ FI0,TH0,SI0,PX0,PY0,PZ0
     &             DT1,DT2,DT3,DT4,DT5,
     &             AL1,AL2,AL3,AL4,AL5,
     &             AA1,AA2,AA3,AA4,AA5,
     &             DD1,DD2,DD3,DD4,DD5,
     &             BL1,BL2,BL3,BL4,BL5,
     &             DF6,TH6,SI6,PX6,PY6,PZ6
```

```
C Initialize the TIMAT matrix to an I matrix:

      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

      SCALE=100.0

C Initialize the T matrix to an I matrix

      DO II = 1,4
      DO JJ = 1,4
        T(II,JJ) = TIMAT(II,JJ)
      ENDDO
      ENDDO

C Manipulator joint angles

      TH1 = DT1 + X(1)
      TH2 = DT2 + X(2)
      TH3 = DT3 + X(3)
      TH4 = DT4 + X(4)
      TH5 = DT5 + X(5)
      FI6 = DF6 + X(6)

C Compute the T matrices, T1 thru T6:

      CALL T3RPY (FI0,TH0,SI0,TRPY)
      CALL T3XYZ (PX0,PY0,PZ0,TXYZ)
      CALL MATMULC (T0,TRPY,TXYZ)

      CALL TRANSFORM ( AL1, AA1, DD1, TH1, BL1, T1 )
      CALL TRANSFORM ( AL2, AA2, DD2, TH2, BL2, T2 )
      CALL TRANSFORM ( AL3, AA3, DD3, TH3, BL3, T3 )
      CALL TRANSFORM ( AL4, AA4, DD4, TH4, BL4, T4 )
      CALL TRANSFORM ( AL5, AA5, DD5, TH5, BL5, T5 )

      CALL T3RPY ( FI6, TH6, SI6, TRPY )
      CALL T3XYZ ( PX6, PY6, PZ6, TXYZ )
      CALL MATMULC ( T6, TRPY, TXYZ )

C Compute the overall transformation, T:

      CALL MATMULA ( T, T0 )
      CALL MATMULA ( T, T1 )
      CALL MATMULA ( T, T2 )
      CALL MATMULA ( T, T3 )
      CALL MATMULA ( T, T4 )
      CALL MATMULA ( T, T5 )
      CALL MATMULA ( T, T6 )

C Calculate the function F
```

```
      F(1)=T(1,4)-TDES(1)
      F(2)=T(2,4)
      F(3)=T(3,4)

C Calculate residual and write to screen

      SUM = 0.0
      DO i=1,3
        XSSQ=SUM+F(I)
      ENDDO

      WRITE (6,*) XSSQ

      RETURN
      END

C ***************************************************************

      SUBROUTINE RANDOM (X,Z)

C This subroutine generates random numbers in the range 0-1
C using a supplied seed x, the returned random number being z.

      REAL FM, FX, Z
      INTEGER A, X, I, M
      DATA I/1/

      IF ( I .EQ. 0 ) GO TO 1000
      I=0
      M= 2 ** 20
      FM= M
      A= 2**10 + 3
 1000 X= MOD( A*X ,M)
      FX= X
      Z= FX/ FM

      RETURN
      END
```

APPENDIX B

```
      PROGRAM BID6

C Robot Identification using the Non-linear Least Squares method for the modified linear slide method.
C Simulation data is read for the PUMA manipulator from the data file PUMA-POS.DAT

C Change parameter LDFJAC to change the number of observations,
C set LDFJAC = 6 * Number of observations

      INTEGER LDFJAC, MM, M, NN, N, NSIG, MAXFN, IOPT, IXJAC, INFER, IER
      PARAMETER (LDFJAC=3*100, MM=LDFJAC, NN=26)

      REAL*8 FJAC(LDFJAC,NN), XJTJ((NN+1)*NN/2)
      REAL*8 PARM(4), F(LDFJAC), WORK((5*NN)+(2*MM)+((NN+1)*NN/2))
      REAL*8 X(NN)
      EXTERNAL PUMA_ARM

      REAL*8 DANGLE, DLENTH, TQ, DQ, EPS, DELTA, SSQ
      REAL*8 SQERR1, SQERR2

      REAL*8 FI0,TH0,SI0,PX0,PY0,PZ0
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, TH6, SI6, PX6, PY6, PZ6, FI6

      INTEGER I, J, K, NOBS, MAXNOBS
      PARAMETER (MAXNOBS=360)
      REAL*8 TET1(MAXNOBS), TET2(MAXNOBS), TET3(MAXNOBS)
      REAL*8 TET4(MAXNOBS), TET5(MAXNOBS), TET6(MAXNOBS)
      REAL*8 TM(3,MAXNOBS), SCALE
      COMMON /PDATA/ NOBS, TM, SCALE,
     &TET1, TET2, TET3, TET4, TET5, TET6

C Open data files for inputs and results

      OPEN (8, NAME='RESULT.DAT', STATUS='NEW')
      OPEN (9, NAME='PUMA-POS.DAT', STATUS='OLD')
      OPEN (10,NAME='INPUT.DAT', STATUS='OLD')

c Read input parameters

      READ (10,*)
      READ (10,*) FI0,TH0,SI0,PX0,PY0,PZ0
```

```
      READ (10,*) DT1,DD1,AA1,AL1,BL1
      READ (10,*) DT2,DD2,AA2,AL2,BL2
      READ (10,*) DT3,DD3,AA3,AL3,BL3
      READ (10,*) DT4,DD4,AA4,AL4,BL4
      READ (10,*) DT5,DD5,AA5,AL5,BL5
      READ (10,*)
      READ (10,*) DF6,TH6,SI6,PX6,PY6,PZ6
      READ (10,*)
      READ (10,*) NOBS,N,DANGLE,DLENTH,MAGNX,NAGN1

      CLOSE (10)

C Initialize data variables

      X(1)=FI0
      X(2)=TH0
      X(3)=PX0
      X(4)=PY0
      x(5)=PZ0

      x(6)=AA1
      X(7)=AL1

      X(8)=DT2
      X(9)=AA2
      X(10)=AL2
      X(11)=BL2

      X(12)=DT3
      X(13)=DD3
      X(14)=AA3
      X(15)=AL3

      X(16)=DT4
      X(17)=DD4
      X(18)=AA4
      X(19)=AL4

      X(20)=DT5
      X(21)=DD5
      X(22)=AA5
      X(23)=AL5

      X(24)=DF6
      X(25)=PX6
      X(26)=PZ6

C Read simulated joint data and tool pose

      DO J = 1, NOBS
        READ (9,*) TET1(J), TET2(J), TET3(J), TET4(J), TET5(J), TET6(J)
```

```
        READ (9,*) TM(1,J), TM(2,J), TM(3,J)
        READ (9,*)
      ENDDO
      CLOSE (9)

C Initialize scale for the angular rows of the Jacobian

      SCALE=100.0

C Call IMSL routine for non-linear identification

      NSIG=4
      EPS=0.0
      DELTA=0.0
      MAXFN=1500
      IOPT=1
      IXJAC=LDFJAC
      M=3*NOBS

      CALL ZXSSQ(PUMA_ARM,M,N,NSIG,EPS,DELTA,MAXFN,IOPT,
     &           PARM,X,SSQ,F,FJAC,IXJAC,XJTJ,WORK,INFER,IER)

C Save results to data file

      WRITE (8,*)
      WRITE (8,*) 'FI0, TH0, SI0, PX0, PY0, PZ0'
      WRITE (8,888) X(1), X(2), 0.0, X(3), x(4), x(5)
      WRITE (8,*)
      WRITE (8,*) 'DT1, DD1, AA1, AL1, BL1'
      WRITE (8,888) 0.0, 0.0, X(6), X(7), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT2, DD2, AA2, AL2, BL2'
      WRITE (8,888) X(8), 0.0, X(9), X(10), X(11)
      WRITE (8,*)
      WRITE (8,*) 'DT3, DD3, AA3, AL3, BL3'
      WRITE (8,888) X(12), X(13), X(14), X(15), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT4, DD4, AA4, AL4, BL4'
      WRITE (8,888) X(16), X(17), X(18), X(19), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT5, DD5, AA5, AL5, BL5'
      WRITE (8,888) X(20), X(21), X(22), X(23), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DF6, TH6, SI6, PX6, PY6, PZ6'
      WRITE (8,889) X(24), 0.0, 0.0, X(25), 0.0, X(26)

888   FORMAT ( 5F12.5 )
889   FORMAT ( 6F12.5 )

C Calculate root mean square error in identification
```

```
      TQ = DANGLE
      DQ = DLENTH

C Error in identification (angular parameters)

      SQERR1 =
     &  (FIO+TQ-X(1))**2 +(SIO+TQ-X(2))**2
     &  +(DT3+TQ-X(12))**2 +(DT4+TQ-X(16))**2 +(DT5+TQ-X(20))**2
     &   +(AL1+TQ-X(8))**2 +(AL2+TQ-X(11))**2
     &  +(AL3+TQ-X(15))**2 +(AL4+TQ-X(19))**2 +(AL5+TQ-X(23))**2
     &  +(BL2+TQ-X(11))**2 +(DT2+TQ-X(8))**2
     &  +(DF6+TQ-X(25))**2
      SQERR1 = DSQRT( SQERR1/13 )

C Error in identification (length parameters)

      SQERR2 =
     &   (PXO+DQ-X(3))**2 +(PYO+DQ-X(4))**2 +(PZO+DQ-X(5))**2
     &   +(AA1+DQ-X(6))**2 +(AA2+DQ-X(9))**2
     &  +(AA3+DQ-X(14))**2 +(AA4+DQ-X(18))**2 +(AA5+DQ-X(22))**2
     &  +(DD3+DQ-X(14))**2 +(DD4+DQ-X(18))**2 +(DD5+DQ-X(22))**2
     &  +(PX6+DQ-X(25))**2 +(PZ6+DQ-X(26))**2
      SQERR2 = DSQRT( SQERR2/13 )

      WRITE (8,*)
      WRITE (8,*) 'RMS PARMS (LENGTH), RMS PARMS (ANGLE)'
      WRITE (8,*) SQERR2, SQERR1
      WRITE (6,*) 'RMS PARMS (LENGTH), RMS PARMS (ANGLE)'
      WRITE (6,*) SQERR2,SQERR1


      WRITE (8,*)
      WRITE (8,*) 'INFER, IER,NOBS,NSIG'
      WRITE (8,*)  INFER, IER,NOBS,NSIG
      WRITE (6,*) 'INFER, IER,NOBS,NSIG'
      WRITE (6,*)  INFER, IER,NOBS,NSIG

      WRITE (8,*)

      CLOSE (8)

      END

C ***************************************************************************

      SUBROUTINE PUMA_ARM (X, M, N, F)

C This subroutine calculates the non-linear function for the use of
C the IMSL routine DUNLSF. It is the forward kinematic solution for
C the PUMA manipulator.
```

```
      INTEGER M, N
      REAL*8 X(N), F(M)
      INTEGER II, JJ
      REAL*8 FI0,TH0,SI0,PX0,PY0,PZ0
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 FI6, TH6, SI6, PX6, PY6, PZ6, DF6

      REAL*8 TH1, TH2, TH3, TH4, TH5
      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4), T4(4,4)
      REAL*8 T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4)
      REAL*8 TINV(4,4), THJ(4,4), TDELTA(4,4)

      INTEGER I, J, K, NOBS, MAXNOBS
      PARAMETER (MAXNOBS=360)
      REAL*8 TET1(MAXNOBS), TET2(MAXNOBS), TET3(MAXNOBS)
      REAL*8 TET4(MAXNOBS), TET5(MAXNOBS), TET6(MAXNOBS)
      REAL*8 TM(3,MAXNOBS), SCALE
      COMMON /PDATA/ NOBS, TM, SCALE,
     &      TET1, TET2, TET3, TET4, TET5, TET6

C Initialize the TIMAT matrix to an I matrix:

      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

C Set parameters for the manipulator:

      FI0=X(1)
      TH0=X(2)
      SI0=0.0
      PX0=X(3)
      PY0=X(4)
      PZ0=x(5)

      DT1=0.0
      DD1=0.0
      AA1=x(6)
      AL1=X(7)
      BL1=0.0

      DT2=X(8)
      DD2=0.0
      AA2=X(9)
      AL2=X(10)
      BL2=X(11)

      DT3=X(12)
```

```
      DD3=X(13)
      AA3=X(14)
      AL3=X(15)
      BL3=0.0

      DT4=X(16)
      DD4=X(17)
      AA4=X(18)
      AL4=X(19)
      BL4=0.0

      DT5=X(20)
      DD5=X(21)
      AA5=X(22)
      AL5=X(23)
      BL5=0.0

      DF6=X(24)
      TH6=0.0
      SI6=0.0
      PX6=X(25)
      PY6=0.0
      PZ6=X(26)

C Loop NOBS times

      K = 0
      DO J = 1, NOBS

C Initialize the T matrix to an I matrix

      DO II = 1,4
      DO JJ = 1,4
        T(II,JJ) = TIMAT(II,JJ)
      ENDDO
      ENDDO

C Manipulator joint angles

      TH1 = DT1 + TET1(J)
      TH2 = DT2 + TET2(J)
      TH3 = DT3 + TET3(J)
      TH4 = DT4 + TET4(J)
      TH5 = DT5 + TET5(J)
      FI6 = DF6 + TET6(J)

C Compute the T matrices, T1 thru T6:

      CALL T3RPY ( FI0, TH0, SI0, TRPY)
      CALL T3XYZ ( PX0, PY0, PZ0, TXYZ)
      CALL MATMULC ( T0, TRPY, TXYZ)
```

```
      CALL TRANSFORM ( AL1, AA1, DD1, TH1, BL1, T1 )
      CALL TRANSFORM ( AL2, AA2, DD2, TH2, BL2, T2 )
      CALL TRANSFORM ( AL3, AA3, DD3, TH3, BL3, T3 )
      CALL TRANSFORM ( AL4, AA4, DD4, TH4, BL4, T4 )
      CALL TRANSFORM ( AL5, AA5, DD5, TH5, BL5, T5 )


      CALL T3RPY ( FI6, TH6, SI6, TRPY )
      CALL T3XYZ ( PX6, PY6, PZ6, TXYZ )
      CALL MATMULC (T6, TRPY, TXYZ )

C Compute the overall transformation, T:

      CALL MATMULA ( T, T0 )
      CALL MATMULA ( T, T1 )
      CALL MATMULA ( T, T2 )
      CALL MATMULA ( T, T3 )
      CALL MATMULA ( T, T4 )
      CALL MATMULA ( T, T5 )
      CALL MATMULA ( T, T6 )

C Get the "T-measured" matrix for this observation:

      DO II = 1,3
       DO JJ = 1,4
        TMJ(II,JJ) = TM(II,JJ,J)
       ENDDO
      ENDDO

      TMJ(4,1) = 0.0
      TMJ(4,2) = 0.0
      TMJ(4,3) = 0.0
      TMJ(4,4) = 1.0

C Calculate the functions F

      K = K + 1
      F(K) = T(1,4)-TM(1,J)
      K = K + 1
      F(K) = T(2,4)-TM(2,J)
      K = K + 1
      F(K) = T(3,4)-TM(3,J)

C End the do-loop for counter J

      ENDDO

C Write RMS error in F

        XSSQ=0.0
        DO II=1,3*NOBS
```

```
        XSSQ=XSSQ+F(II)*F(II)
      ENDDO

      XER=SQRT(XSSQ)
      WRITE(6,*) XER

    RETURN
    END

C ***************************************************************************
```

APPENDIX C

```
C ************************************************************************

      PROGRAM VERIFY

C This program generates the six-dof pose  error for the PUMA manipulator.
C It contains the identified calibration parameters and the exact parameter.
C It uses a data file of verification joint angle sets POSEVER.DAT, and the
C file RESULT.DAT from the program ID6.

      INTEGER I, J, K, NPOSES, N
      REAL*8 DANGLE, DLENTH
      REAL*8 DT(5),DD(5),AA(5),AL(5),BL(5),MEAS(6)
      REAL*8 EDT(5),EDD(5),EAA(5),EAL(5),EBL(5),EMEAS(6)
      REAL*8 EDF6,EFI6,ETH6,ESI6,EPX6,EPY6,EPZ6
      REAL*8 THETA(1000,6), TDELTA(4,4)
      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4)
      REAL*8 T4(4,4), T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4), ET(4,4)

      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6
      REAL*8 XW, YW, ZW
      COMMON TIMAT,THETA

C Initialize the TIMAT matrix to an I matrix:

      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

C Open data file

      OPEN (9,  NAME='POSEVER.DAT',STATUS='OLD')
      OPEN (10, NAME='INPUT.DAT', STATUS='OLD')
      OPEN (11, NAME='RESULT.DAT', STATUS='OLD')

C Read input parameters

      READ (10,*)
      READ (10,*) MEAS(1),MEAS(2),MEAS(3),MEAS(4),MEAS(5),MEAS(6)
      READ (10,*) DT1,DD1,AA1,AL1,BL1
      READ (10,*) DT2,DD2,AA2,AL2,BL2
      READ (10,*) DT3,DD3,AA3,AL3,BL3
```

```
      READ (10,*) DT4,DD4,AA4,AL4,BL4
      READ (10,*) DT5,DD5,AA5,AL5,BL5
      READ (10,*
      READ (10,*) DF6,TH6,SI6,PX6,PY6,PZ6
      READ (10,*)
      READ (10,*) NOBS,R,DANGLE,DLENTH,MAGNX,MAGNL

      CLOSE (10)


C Read in joint angle sets for verification poses

      NPOSES=NOBS

      DO I=1,NPOSES
        READ(9,*)
        READ(9,*)THETA(I,1),THETA(I,2),THETA(I,3),THETA(I,4),
     &           THETA(I,5),THETA(I,6)
      ENDDO
      CLOSE(9)

C Set exact link parameters for the manipulator:

      DO I=2,5
        DT(I)=DT(I)+DANGLE
      ENDDO

      MEAS(1)=MEAS(1)+DLENTH
      MEAS(2)=MEAS(2)+DLENTH
      MEAS(3)=MEAS(3)+DLENTH
      MEAS(4)=MEAS(4)+DLENTH
      MEAS(5)=MEAS(5)+DLENTH
      MEAS(6)=MEAS(6)+DLENTH

      AL(1)=AL1+DANGLE
      AL(2)=AL2+DANGLE
      AL(3)=AL3+DANGLE
      AL(4)=AL4+DANGLE
      AL(5)=AL5+DANGLE

      AA(1) = AA1 + DLENTH
      AA(2) = AA2 + DLENTH
      AA(3) = AA3 + DLENTH
      AA(4) = AA4 + DLENTH
      AA(5) = AA5 + DLENTH

      DD(1) = DD1
      DD(2) = DD2
      DD(3) = DD3 + DLENTH
      DD(4) = DD4 + DLENTH
      DD(5) = DD5 + DLENTH
```

```
      BL(1) = BL1
      BL(2) = BL2 + DANGLE
      BL(3) = BL3
      BL(4) = BL4
      BL(5) = BL4

      DF6 = DF6 + DANGLE
      TH6 = TH6
      SI6 = SI6
      PX6 = PX6 + DLENTH
      PY6 = PY6
      PZ6 = PZ6 + DLENTH

C Read in and set up estimated parameter table

      READ(11,*)
      READ(11,*)
      READ(11,*) EMEAS(1),EMEAS(2),EMEAS(3)

      DO I=1,5
        READ(11,*)
        READ(11,*)
        READ (11,*) EDT(I),EDD(I),EAA(I),EAL(I),EBL(I)
      ENDDO

      READ(11,*)
      READ(11,*)
      READ(11,*) EDF6,ETH6,ESI6,EPX6,EPY6,EPZ6

C Main loop through NPOSES joint angle sets

      DO K=1,NPOSES

        CALL FKS (K,MEAS,DT,AL,AA,DD,BL,FI6,TH6,SI6,PX6,PY6,PZ6,T)
        CALL FKS (K,EMEAS,EDT,EAL,EAA,EDD,EBL,EFI6,ETH6,ESI6,EPX6,
     &            EPY6,EPZ6,ET)

C Compute the differential tool matrix

        CALL MATSUB(TDELTA,T,ET)

c Compute the pose errors

        POSERR=SQRT(TDELTA(1,4)**2+TDELTA(2,4)**2+TDELTA(3,4)**2)
        ORERR1=(TDELTA(3,2)-TDELTA(2,3))/2
        ORERR2=(TDELTA(1,3)-TDELTA(3,1))/2
        ORERR3=(TDELTA(2,1)-TDELTA(1,2))/2
        ORERR=SQRT(ORERR1**2+ORERR2**2+ORERR3**2)

c Update total error counts
```

```
        POSTERR=(POSERR+(K-1)*POSTERR)/K
        ORTERR=(ORERR+(K-1)*ORTERR)/K

c End of main loop

      ENDDO

      WRITE(6,*) 'Position error, orientation error'
      WRITE(6,*) POSTERR,ORTERR
      END

c ****************************************************************

      SUBROUTINE FKS (N,MEAS,DT,AL,AA,DD,BL,DF6,TH6,SI6,
     &                PX6,PY6,PZ6,T)

      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4)
      REAL*8 T4(4,4), T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4), dt(5),al(5),aa(5),dd(5),bl(5)
      REAL*8 THETA(1000,6),ANG(5),MEAS(6)

      COMMON TIMAT,THETA

C Initialize the T matrix to an I matrix:

      DO J=1,4
        DO K=1,4
          T(J,K) = TIMAT(J,K)
        ENDDO
      ENDDO

C Set up the joint angles

      DO I=1,5
        ANG(I)=THETA(N,I)+DT(I)
      ENDDO

      FI6=THETA(N,6)+DF6

C Compute the T matrices, T1 thru T6:

      CALL T3RPY (MEAS(1),MEAS(2),MEAS(3),T0)
      CALL T3XYZ (MEAS(4),MEAS(5),MEAS(6),T0)
      CALL MATMULC (T0,TRPY,TXYZ)

      CALL TRANSFORM (AL(1),AA(1),DD(1),ANG(1),BL(1),T1)
      CALL TRANSFORM (AL(2),AA(2),DD(2),ANG(2),BL(2),T1)
      CALL TRANSFORM (AL(3),AA(3),DD(3),ANG(3),BL(3),T1)
      CALL TRANSFORM (AL(4),AA(4),DD(4),ANG(4),BL(4),T1)
      CALL TRANSFORM (AL(5),AA(5),DD(5),ANG(5),BL(5),T1)
```

```
      CALL T3RPY (FI6,TH6,SI6,TRPY )
      CALL T3XYZ (PX6,PY6,PZ6,TXYZ )
      CALL MATMULC (T6,TRPY,TXYZ )

C Compute the overall transformation, T:

      CALL MATMULA ( T, T0 )
      CALL MATMULA ( T, T1 )
      CALL MATMULA ( T, T2 )
      CALL MATMULA ( T, T3 )
      CALL MATMULA ( T, T4 )
      CALL MATMULA ( T, T5 )
      CALL MATMULA ( T, T6 )

      RETURN
      END

C ***************************************************************************
```

APPENDIX D

```
      PROGRAM WIRE

C This program generates a set of joint angles for the calibration
C of the PUMA manipulator using a wire potentiometer attached to
C the end point of the manipulator.

      INTEGER LDFJAC, M, N, obs, nobs
      PARAMETER (LDFJAC=6, M=LDFJAC, N=6)

      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6
      REAL*8 XW, YW, ZW
      REAL*8 RUXYZ(3),UXYZ(3),XYZ,DALPHA,DBETA,DGAMMA,ANUM
      REAL*8 PDIRCOS,NDIRCOS,PNGLBTWN,NNGLBTWN

      INTEGER INFER,IER,IOPT,NSIG,MAXFN
      REAL*8 FJAC(LDFJAC,N), XJTJ((N+1)*N/2), XJAC(LDFJAC,N)
      REAL*8 PARM(4),F(LDFJAC), WORK((5*N)+(2*M)+((N+1)*N/2))
      REAL*8 X(N),XD,YD,AA
      REAL*8 R,PHIMAX,PHIMIN,THETAMAX,THETAMIN,PHI,THETA
      REAL*8 XB,YB,ZB,SSQ,RR,MAGNX,MAGN1,QQ,PI
      REAL*8 RAD,TX,GAMMA,DPSI,DPHI,OT,OTX

      EXTERNAL PUMA_ARM

      REAL*8 OTTPOP,OOP,T6(4,4)
      INTEGER I, J, K
      REAL*8 TDES(4,4), QMAX(6), QMIN(6), SCALE, DANGLE, DLENTH, NUM
      COMMON/LEN/ PI,R,T6,THETAU,THETAL,TTP
      COMMON /PDATA/ DANGLE, DLENTH,TDES
      COMMON /KIN/ DT1,DT2,DT3,DT4,DT5,
     &             AL1,AL2,AL3,AL4,AL5,
     &             AA1,AA2,AA3,AA4,AA5,
     &             DD1,DD2,DD3,DD4,DD5,
     &             BL1,BL2,BL3,BL4,BL5,
     &             XW,YW,ZW,
     &             DF6,TH6,SI6,PX6,PY6,PZ6

      RAD=25.40d0/2.0d0
      PI=4.0d0*DATAN(1.0D0)
```

```
C Initialize data variables

      OBS=0

C Open data files for input


      OPEN (10, NAME='PUMA-SOLN.DAT', STATUS='NEW')
      OPEN (9,  NAME='INPUT.DAT', STATUS='OLD')

C Read input kinematic data

      READ (9,*)
      READ (9,*) XW,YW,ZW
      READ (9,*) DT1,DD1,AA1,AL1,BL1
      READ (9,*) DT2,DD2,AA2,AL2,BL2
      READ (9,*) DT3,DD3,AA3,AL3,BL3
      READ (9,*) DT4,DD4,AA4,AL4,BL4
      READ (9,*) DT5,DD5,AA5,AL5,BL5
      READ (9,*)
      READ (9,*) DF6,TH6,SI6,PX6,PY6,PZ6
      READ (9,*)
      READ (9,*) NOBS,QP,DANGLE,DLENTH,MAGNX,MAGN1

      CLOSE (9)

C Adjust nominal values

      XW=XW+DLENTH
      YW=YW+DLENTH

      DT2=DT2+DANGLE
      DT3=DT3+DANGLE
      DT4=DT4+DANGLE
      DT5=DT5+DANGLE

      AL1=AL1+DANGLE
      AL2=AL2+DANGLE
      AL3=AL3+DANGLE
      AL4=AL4+DANGLE
      AL5=AL5+DANGLE

      AA1=AA1+DLENTH
      AA2=AA2+DLENTH
      AA3=AA3+DLENTH
      AA4=AA4+DLENTH
      AA5=AA5+DLENTH

      DD1=DD1+DLENTH
      DD3=DD3+DLENTH
      DD4=DD4+DLENTH
```

```
      DD5=DD5+DLENTH

      BL2=BL2+DANGLE

      DF6=DF6+DANGLE
      PX6=PX6+DLENTH
      PZ6=PZ6+DLENTH

C Set limits on spherical coordinates

      PHIMAX=90.0
      PHIMIN=0.0
      THETAMAX=0.0
      THETAMIN=360.0

C Get random number seed

      WRITE (6,*) 'Type in a 6-digit random number seed'
      READ (5,*) ISEED

C Start of main loop

 1010 OBS=OBS+1

C Set joint angles to zero

        DO I=1,N
           X(I)=0.0D0
        ENDDO

C Get random spherical coordinats for end effector

 1000   CALL RANDOM (ISEED,NUM)
        PHI=PHIMIN+(PHIMAX-PHIMIN)*NUM
        CALL RANDOM (ISEED,NUM)
        THETA=THETAMIN+(THETAMAX-THETAMIN)*NUM
        CALL RANDOM (ISEED,NUM)
        Q=100.0+900.0*NUM
C Calculate end point of the manipulator

        XB=Q*COSD(THETA)*SIND(PHI)
        YB=Q*SIND(THETA)*SIND(PHI)
        ZB=Q*COSD(PHI)

        IF (ZB .LT. 50.0) GOTO 1000

C Calculate unit vector between base and end effector

        XYZ=DSQRT(XB**2+YB**2+ZB**2)
        UXYZ(1)=-XB/XYZ
        UXYZ(2)=-YB/XYZ
```

```
      UXYZ(3)=-ZB/XYZ

C Calculate direction angles from direction cosines

      DALPHA=DACOS(UXYZ(1))
      DBETA=DACOS(UXYZ(2))
      DGAMMA=DACOS(UXYZ(3))

C Perturb direction angles

 33   CALL RANDOM(ISEED,NUM)
      ANUM=(0.50D0-NUM)*PI/6.0D0
      RUXYZ(1)=DCOS(DALPHA+ANUM)
      CALL RANDOM(ISEED,NUM)
      ANUM=(0.50D0-NUM)*PI/6.0D0
      RUXYZ(3)=DCOS(DGAMMA+ANUM)
      CHECK=RUXYZ(1)**2+RUXYZ(3)**2
      IF (CHECK .GT. 1.0D0) GOTO 33
      PDIRCOS=DSQRT(1.0D0-RUXYZ(1)**2-RUXYZ(3)**2)
      NDIRCOS=-PDIRCOS
      PNGLBTWN=DACOS(UXYZ(1)*RUXYZ(1)+UXYZ(2)*PDIRCOS+UXYZ(3)
     &         *RUXYZ(3))
      NNGLBTWN=DACOS(UXYZ(1)*RUXYZ(1)+UXYZ(2)*NDIRCOS+UXYZ(3)
     &         *RUXYZ(3))
      RUXYZ(2)=PDIRCOS
      IF (DABS(PNGLBTWN) .GT. DABS(NNGLBTWN)) RUXYZ(2)=NDIRCOS

C Establish desired tool pose

      DO II=1,4
         DO JJ=1,4
            TDES(ii,jj)=0.0
         ENDDO
      ENDDO

      TDES(1,3)=RUXYZ(1)
       TDES(2,3)=RUXYZ(2)
       TDES(3,3)=RUXYZ(3)
      TDES(1,4)=XB
      TDES(2,4)=YB
      TDES(3,4)=ZB
      TDES(4,4)=1.0

C Call IMSL ZXSSQ for inverse kinematic solution

      NSIG=4
      EPS=0.0
      DELTA=0.0
      MAXFN=500
      IOPT=1
      IXJAC=LDFJAC
```

```
        CALL ZXSSQ(PUMA_ARM,M,N,NSIG,EPS,DELTA,MAXFN,IOPT,PARM,X,
     &                  SSQ,F,XJAC,IXJAC,XJTJ,WORK,INFER,IER)

C Check for singularities

        IF (SSQ .GT. 0.00001) GOTO 1000

C Compute wire length

         CALL LENGTH(OTTPOP)

C Inject noise on wire length

         CALL RANDOM(ISEED,NUM)
         OTTPOP=OTTPOP+((0.5-NUM)*2.0)*MAGNX

C Write simulation data to file

         WRITE(10,*)OTTPOP
        WRITE (10,888) X(1), X(2), X(3), X(4), X(5), X(6)
 888     FORMAT ( 6F12.3 )

C Continue for other end effector positions

      IF (OBS .LT. NOBS) GOTO 1010

      CLOSE (10)
      END

C ***************************************************************************

      SUBROUTINE PUMA_ARM (X,M,N,F)

C This subroutine calculates the non-linear function for the use of
C the IMSL routine ZXSSQ. It is the forward kinematic solution for
C the PUMA manipulator.

      INTEGER M, N
      REAL*8 X(N), F(M)

      INTEGER II, JJ
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6
      REAL*8 XW, YW, ZW

      REAL*8 TH1, TH2, TH3, TH4, TH5
      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4), T4(4,4)
```

```
      REAL*8 T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4),PI,THETAU,THETAL,TTP
      REAL*8 DISQ,DIS,SUM

      INTEGER I, J, K
      REAL*8 TDES(4,4), DANGLE, DLENTH, R

      COMMON/LEN/ PI,R,T,THETAU,THETAL,TTP
      COMMON /PDATA/DANGLE, DLENTH,TDES
      COMMON /KIN/ DT1,DT2,DT3,DT4,DT5,
     &             AL1,AL2,AL3,AL4,AL5,
     &             AA1,AA2,AA3,AA4,AA5,
     &             DD1,DD2,DD3,DD4,DD5,
     &             BL1,BL2,BL3,BL4,BL5,
     &             XW,YW,ZW,
     &             DF6,TH6,SI6,PX6,PY6,PZ6

C Initialize the TIMAT matrix to an I matrix:

      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

C Initialize the T matrix to an I matrix
      DO II = 1,4
      DO JJ = 1,4
        T(II,JJ) = TIMAT(II,JJ)
      ENDDO
      ENDDO

C Manipulator joint angles

      TH1 = DT1 + X(1)
      TH2 = DT2 + X(2)
      TH3 = DT3 + X(3)
      TH4 = DT4 + X(4)
      TH5 = DT5 + X(5)
      FI6 = DF6 + X(6)

C Compute the T matrices, T1 thru T6:

      CALL T3XYZ ( XW, YW, ZW, T0 )

      CALL TRANSFORM ( AL1, AA1, DD1, TH1, BL1, T1 )
      CALL TRANSFORM ( AL2, AA2, DD2, TH2, BL2, T2 )
      CALL TRANSFORM ( AL3, AA3, DD3, TH3, BL3, T3 )
      CALL TRANSFORM ( AL4, AA4, DD4, TH4, BL4, T4 )
      CALL TRANSFORM ( AL5, AA5, DD5, TH5, BL5, T5 )

      CALL T3RPY ( FI6, TH6, SI6, TRPY )
      CALL T3XYZ ( PX6, PY6, PZ6, TXYZ )
      CALL MATMULC ( T6, TRPY, TXYZ )
```

```
C Compute the overall transformation, T:

      CALL MATMULA ( T, T0 )
      CALL MATMULA ( T, T1 )
      CALL MATMULA ( T, T2 )
      CALL MATMULA ( T, T3 )
      CALL MATMULA ( T, T4 )
      CALL MATMULA ( T, T5 )
      CALL MATMULA ( T, T6 )

C Calculate the function F

      F(1)=T(1,4)-TDES(1,4)
      F(2)=T(2,4)-TDES(2,4)
      F(3)=T(3,4)-TDES(3,4)
      F(4)=(T(1,3)-TDES(1,3))*100.0D0
      F(5)=(T(2,3)-TDES(2,3))*100.0D0
      F(6)=(T(3,3)-TDES(3,3))*100.0D0


C Calculate residual

      SUM=0.0D0
      DO IJKL=1,6
        SUM=SUM+F(IJKL)**2
      ENDDO
      WRITE(6,*)DSQRT(SUM)/6.0D0

      RETURN
      END

C ***************************************************************************

      SUBROUTINE RANDOM (x,z)

C This subroutine generates random numbers in the range 0-1
C using a supplied seed x, the returned random number being z.

      REAL FM, FX, Z
      INTEGER A, X, I, M
      DATA I/1/

      IF ( I .EQ. 0 ) GO TO 1000
      I=0
      M= 2 ** 20
      FM= M
      A= 2**10 + 3

 1000 X= MOD( A*X ,M)
      FX= X
      Z= FX/ FM
```

```
      RETURN
      END

C ***********************************************************************
C
C This subroutine calculates the length of wire between the end effector
C frame and measurement base frame. The subroutine utilizes IMSL routine
C ZXSSQ for solution of the length along with subroutine MINLENTH

      SUBROUTINE LENGTH(OTTPOP)
      REAL*8 T6(4,4),T6INV(4,4)
      REAL*8 EPSN,DELTAN,PARMN(4),XN(4),SSQN,FN(6),XJACN(6,4)
      REAL*8 WORKN(42),XJTJN(10),XUUVP(4)
      INTEGER MN,NN,NSIGN,MAXFNN,IOPTN,IXJACN,INFERN,IERN
      REAL*8 PI,R,X,Y,Z,OOP,XLUV(4),XUUV(4),XO,YO,P6(4),XOYO
      REAL*8 X6Y6,ZO,OTTPOP,OTSCAL,OT,XU(4),TTP,TTPSCAL
      REAL*8 THETAU,THETAL
      COMMON/LEN/ PI,R,T6,THETAU,THETAL,TTP

      EXTERNAL MINLENTH


C Set ZXSSQ parameters

      MN=6
      NN=4
      NSIGN=4
      EPS=0.0D0
      DELTAN=0.0D0
      MAXFNN=1000
      IXJACN=6
      IOPTN=1

      PI=4.0D0*DATAN(1.0D0)
      R=12.70D0

C Calculate initial values for ZXSSQ vector X

      X=T6(1,4)
      Y=T6(2,4)
      Z=T6(3,4)
      OOP=DSQRT(X**2+Y**2+Z**2)

C Calculate unit vector from base frame origin to end effector origin

      XLUV(1)=X/OOP
      XLUV(2)=Y/OOP
      XLUV(3)=Z/OOP
      XLUV(4)=1.0D0

C Zero T6 inverse matrix
```

```
      DO I=1,4
        DO J=1,4
          T6INV(I,J)=0.0D0
        ENDDO
      ENDDO

C Compute inverse of T6 matrix

      DO I=1,3
        DO J=1,3
          T6INV(I,J)=T6(J,I)
        ENDDO
        T6INV(1,4)=T6INV(1,4)-T6(I,4)*T6(I,1)
        T6INV(2,4)=T6INV(2,4)-T6(I,4)*T6(I,2)
        T6INV(3,4)=T6INV(3,4)-T6(I,4)*T6(I,3)
      ENDDO
      T6INV(4,4)=1.0D0

C Calculate coordinates of end effector unit vector in measurement
C frame coordinate

      XUUVP(1)=X-XLUV(1)
      XUUVP(2)=Y-XLUV(2)
      XUUVP(3)=Z-XLUV(3)
      XUUVP(4)=1.0D0

C Convert coordinates to end effector frame reference

      DO I=1,4
        XUUV(I)=0.0D0
        DO J=1,4
          XUUV(I)=XUUV(I)+T6INV(I,J)*(XUUVP(J))
        ENDDO
      ENDDO

C Initialize ZXSSQ X vector

      XN(1)=XLUV(1)*R
      XN(2)=XLUV(2)*R
      XN(3)=XUUV(1)*R
      XN(4)=XUUV(2)*R

C Call ZXSSQ for length calculation

      CALL ZXSSQ(MINLENTH,MN,NN,NSIGN,EPSN,DELTAN,MAXFNN,IOPTN,
     &     PARMN,XN,SSQN,FN,XJACN,IXJACN,XJTJN,WORKN,INFERN,IERN)

      OTTPOP=(THETAL+THETAU)*R+TTP

      RETURN
      END
```

```
C ****************************************************************
C
C
      SUBROUTINE MINLENTH(XN,MN,NN,FN)
      INTEGER MN,NN
      REAL*8 XN(NN),FN(MN),ERROR
      REAL*8 PI,R,X,Y,Z,OOP,XLUV(4),XUUV(4),XO,YO,P6(4),XOYO
      REAL*8 X6Y6,ZO,OPTPSCAL,OPT,OTSCAL,OT,XU(4),TTP,TTPSCAL
      REAL*8 T6(4,4),P6P(4),THETAU,THETAL,MAGUL,MAGUU,UXO
      REAL*8 UYO,UZO,XUP(4),UX6,UY6,UZ6,UXT,UYT,UZT,ZOP
      COMMON/LEN/ PI,R,T6,THETAU,THETAL,TTP

C Initialize variables based on current value of X

      XO=XN(1)
      YO=XN(2)
      P6(1)=XN(3)
      P6(2)=XN(4)
      P6P(1)=P6(1)
      P6P(2)=P6(2)

C Calculate length in each frames xy plane

      XOYO=DSQRT(XO**2+YO**2)
      X6Y6=DSQRT(P6(1)**2+P6(2)**2)


C Calculate corresponding z value

      P6(3)=DSQRT(2.0DO*R*X6Y6-P6(1)**2-P6(2)**2)
      ZO=DSQRT(2.0DO*R*XOYO-XO**2-YO**2)

C Calculate angle theta for arclength calculations

      THETAU=DASIN(P6(3)/R)
      THETAL=DASIN(ZO/R)

C Calculate intermediate z value for unit tangent vectors

      P6P(3)=X6Y6*DTAN(PI/2.0DO-THETAU)
      ZOP=XOYO*DTAN(PI/2.0DO-THETAL)
      MAGUL=DSQRT(XO**2+YO**2+ZOP**2)
      MAGUU=DSQRT(P6P(1)**2+P6P(2)**2+P6P(3)**2)
      UXO=XO/MAGUL
      UYO=YO/MAGUL
      UZO=ZOP/MAGUL

C Transform unit vector in FE coordinates to FM coordinates

      P6P(4)=1.0DO
      DO I=1,4
```

```
        XUP(I)=0.0D0
        DO J=1,4
          XUP(I)=XUP(I)+T6(I,J)*P6P(J)
        ENDDO
      ENDDO

C Transform FE tangent point to FM coordinates

      UX6=(XUP(1)-T6(1,4))/MAGUU
      UY6=(XUP(2)-T6(2,4))/MAGUU
      UZ6=(XUP(3)-T6(3,4))/MAGUU
      P6(4)=1.0D0
      DO I=1,4
        XU(I)=0.0D0
        DO J=1,4
          XU(I)=XU(I)+T6(I,J)*P6(J)
        ENDDO
      ENDDO

C Calculate tangent to tangent distance

      TTP=DSQRT((XU(1)-X0)**2+(XU(2)-Y0)**2+(XU(3)-Z0)**2)

C Calculate tangent point to tangent point unit vector

      UXT=(XU(1)-X0)/TTP
      UYT=(XU(2)-Y0)/TTP
      UZT=(XU(3)-Z0)/TTP

C Calculate minimizing functions


      FN(1)=UXT-UX0
      FN(2)=UYT-UY0
      FN(3)=UZT-UZ0
      FN(4)=UXT+UX6
      FN(5)=UYT+UY6
      FN(6)=UZT+UZ6

C Calculate residual and write to screen

      ERROR =0.0D0
      DO I=1,6
        ERROR=ERROR + FN(I)**2
      ENDDO
      WRITE(6,*)ERROR
      RETURN
      END
```

APPENDIX E

```
      PROGRAM WID6

C Robot Identification using the Non-linear Least Squares method.
C This version of program ID6 is for the wire potentiometer method.
C Simulation data is read for the PUMA manipulator from
C the data file PUMA-SOLN.DAT
C
C SET LDFJAC = NUMBER OF OBSERVATIONS

      INTEGER LDFJAC, MM, M, NN, N, NSIG, MAXFN, IOPT, IXJAC, INFER, IER
      INTEGER I, J, K, NOBS, MAXNOBS
      PARAMETER (LDFJAC=55, MM=LDFJAC, NN=24)
      PARAMETER (MAXNOBS=200)
      REAL*8 FJAC(LDFJAC,NN), XJTJ((NN+1)*NN/2)
      REAL*8 PARM(4), F(LDFJAC), WORK((5*NN)+(2*MM)+((NN+1)*NN/2))
      REAL*8 X(NN),XD,YD,TX,DPSI,GAMMA,DPHI,OT,RAD
      REAL*8 DANGLE, DLENTH, TQ, DQ, EPS, DELTA, SSQ
      REAL*8 SQERR1, SQERR2, PI
      REAL*8 XW,YW,ZW
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 PI6, DF6, TH6, SI6, PX6, PY6, PZ6
      REAL*8 MAGNX,MAGN1
      REAL*8 TET1(MAXNOBS), TET2(MAXNOBS), TET3(MAXNOBS)
      REAL*8 TET4(MAXNOBS), TET5(MAXNOBS), TET6(MAXNOBS)
      REAL*8 R,OTX(MAXNOBS)
      COMMON /PDATA/ NOBS,TET1,TET2,TET3,TET4,TET5,TET6,OTX,RAD

        COMMON /KIN/ DT1,DT2,DT3,DT4,DT5,
     &          AL1,AL2,AL3,AL4,AL5,
     &          AA1,AA2,AA3,AA4,AA5,
     &          DD1,DD2,DD3,DD4,DD5,
     &          BL1,BL2,BL3,BL4,BL5,
     &          XW,YW,ZW,
     &          DF6,TH6,SI6,PX6,PY6,PZ6,PI

      EXTERNAL PUMA_ARM

C Open data files for inputs and results

      OPEN (8, NAME='RESULT.DAT', STATUS='NEW')
```

```
      OPEN (9, NAME='PUMA-SOLN.DAT', STATUS='OLD')
      OPEN (10,NAME='INPUT.DAT', STATUS='OLD')

      PI=4.0D0*DATAN(1.0D0)
      RAD=12.70D0

c Read input parameters

      READ (10,*)
      READ (10,*)XW,YW,ZW
      READ (10,*) DT1,DD1,AA1,AL1,BL1
      READ (10,*) DT2,DD2,AA2,AL2,BL2
      READ (10,*) DT3,DD3,AA3,AL3,BL3
      READ (10,*) DT4,DD4,AA4,AL4,BL4
      READ (10,*) DT5,DD5,AA5,AL5,BL5
      READ (10,*)
      READ (10,*) DF6,TH6,SI6,PX6,PY6,PZ6
      READ (10,*)
      READ (10,*) NOBS,QPR,DANGLE,DLENTH,MAGNX,MAGN1

      CLOSE (10)

C Initialize data variables

      X(1)=XW
      X(2)=YW
      X(3)=XW

      X(4)=AA1
      X(5)=AL1

      X(6)=DT2
      X(7)=AA2
      X(8)=AL2
      X(9)=BL2

      X(10)=DT3
      X(11)=DD3
      X(12)=AA3
      X(13)=AL3

      X(14)=DT4
      X(15)=DD4
      X(16)=AA4
      X(17)=AL4

      X(18)=DT5
      X(19)=DD5
      X(20)=AA5
      X(21)=AL5
```

```
      X(22)=DF6
      X(23)=PX6
      X(24)=PZ6

C Read joint data and wire length.

      DO J = 1, NOBS
        READ (9,*) TET1(J), TET2(J), TET3(J), TET4(J), TET5(J), TET6(J)
          READ (9,*) OTX(J)
          OTX(J)=OTX(J)+25.40D0
          READ (9,*)
      ENDDO
      CLOSE (9)

C Set parameters for IMSL routine ZXSSQ for non-linear identification

      NSIG=4
      EPS=0.0
      DELTA=0.0
      MAXFN=1000
      IOPT=1
      IXJAC=LDFJAC
      M=NOBS

      CALL ZXSSQ(PUMA_ARM,MM,NN,NSIG,EPS,DELTA,MAXFN,IOPT,
     &             PARM,X,SSQ,F,FJAC,IXJAC,XJTJ,WORK,INFER,IER)

C Save results to data file "RESULT.DAT"

      WRITE (8,*)
      WRITE (8,*) 'XW, YW, ZW'
      WRITE (8,93) X(1), X(2), X(3)
      WRITE (8,*)
      WRITE (8,*) 'DT1, DD1, AA1, AL1, BL1'
      WRITE (8,93) 0.0, 0.0, X(4), X(5), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT2, DD2, AA2, AL2, BL2'
      WRITE (8,93) X(6), 0.0, X(7), X(8), X(9)
      WRITE (8,*)
      WRITE (8,*) 'DT3, DD3, AA3, AL3, BL3'
      WRITE (8,93) X(10), X(11), X(12), X(13), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT4, DD4, AA4, AL4, BL4'
      WRITE (8,93) X(14), X(15), X(16), X(17), 0.0
      WRITE (8,*)
      WRITE (8,*) 'DT5, DD5, AA5, AL5, BL5'
      WRITE (8,93) X(18), X(19), X(20), X(21), 0.0
      WRITE (8,*)
      WRITE (8,*) ' DF6, TH6, SI6,  PX6, PY6, PZ6'
      WRITE (8,93)  X(22), 0.0, 0.0, x(23), 0.0, x(24)
 93   FORMAT(2X,6(1X,F10.4))
```

```
      WRITE (8,*)
      WRITE (8,*) 'INFER, IER,NOBS,NSIG'
      WRITE (8,*)  INFER, IER,NOBS,NSIG
      WRITE (6,*) 'INFER, IER,NOBS,NSIG'
      WRITE (6,*)  INFER, IER,NOBS,NSIG
      WRITE (8,*)

      CLOSE (8)

      END

C *********************************************************************

      SUBROUTINE PUMA_ARM (X, M, N, F)

C This subroutine calculates the non-linear function for the use of
C the IMSL routine ZXSSQ. It is the forward kinematic solution for
C the PUMA manipulator.

      INTEGER M, N
      INTEGER II, JJ
      INTEGER I, J, K, NOBS, MAXNOBS
      PARAMETER (MAXNOBS=200)
      REAL*8 X(N), F(M)
      REAL*8 XW, YW, ZW
      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 FI6, DF6, TH6, SI6, PX6, PY6, PZ6
      REAL*8 TH1, TH2, TH3, TH4, TH5
      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4), T4(4,4)
      REAL*8 T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4),XT,YT,ZT,XC,YC,AA,Q
      REAL*8 XD,YD,DPSI,RAD,GAMMA,DPHI,OT,TX,OX
      REAL*8 TET1(MAXNOBS), TET2(MAXNOBS), TET3(MAXNOBS)
      REAL*8 TET4(MAXNOBS), TET5(MAXNOBS), TET6(MAXNOBS)
      REAL*8 RR ,OTX(MAXNOBS),PIK,rms,sumsq
      REAL*8 OTTPOP,PI,R,OOP,THETAU,THETAL,TTP

      COMMON /PDATA/ NOBS,TET1,TET2,TET3,TET4,TET5,TET6,OTX,RAD

      COMMON /KIN/ DT1,DT2,DT3,DT4,DT5,
     &          AL1,AL2,AL3,AL4,AL5,
     &          AA1,AA2,AA3,AA4,AA5,
     &          DD1,DD2,DD3,DD4,DD5,
     &          BL1,BL2,BL3,BL4,BL5,
     &          XW,YW,ZW,
     &          DF6,TH6,SI6,PX6,PY6,PZ6,PI
```

```
C Initialize the TIMAT matrix to an I matrix:

      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

      PIK=PI
C Set parameters for the manipulator:

      XW = X(1)
      YW = X(2)
      ZW = X(3)

      AA1 = X(4)
      AL1 = X(5)

      DT2 = X(6)
      AA2 = X(7)
      AL2 = X(8)
      BL2 = X(9)

      DT3 = X(10)
      DD3 = X(11)
      AA3 = X(12)
      AL3 = X(13)

      DT4 = X(14)
      DD4 = X(15)
      AA4 = X(16)
      AL4 = X(17)

      DT5 = X(18)
      DD5 = X(19)
      AA5 = X(20)
      AL5 = X(21)

      DF6 = X(22)
      PX6 = X(23)
      PZ6 = X(24)

C Loop NOBS times

      K = 0
      DO J = 1, NOBS

C Initialize the T matrix to an I matrix

        DO II = 1,4
          DO JJ = 1,4
            T(II,JJ) = TIMAT(II,JJ)
          ENDDO
        ENDDO
```

```
C Manipulator joint angles

        TH1 = DT1 + TET1(J)
        TH2 = DT2 + TET2(J)
        TH3 = DT3 + TET3(J)
        TH4 = DT4 + TET4(J)
        TH5 = DT5 + TET5(J)
        FI6 = DF6 + TET6(j)

C Compute the T matrices, T1 thru T6:

        CALL T3XYZ (XW,YW,ZW,T0)

        CALL TRANSFORM ( AL1, AA1, DD1, TH1, BL1, T1 )
        CALL TRANSFORM ( AL2, AA2, DD2, TH2, BL2, T2 )
        CALL TRANSFORM ( AL3, AA3, DD3, TH3, BL3, T3 )
        CALL TRANSFORM ( AL4, AA4, DD4, TH4, BL4, T4 )
        CALL TRANSFORM ( AL5, AA5, DD5, TH5, BL5, T5 )

        CALL T3RPY ( FI6, TH6, SI6, TRPY )
        CALL T3XYZ ( PX6, PY6, PZ6, TXYZ )
        CALL MATMULC ( T6, TRPY, TXYZ )

C Compute the overall transformation, T:

        CALL MATMULA ( T, T0 )
        CALL MATMULA ( T, T1 )
        CALL MATMULA ( T, T2 )
        CALL MATMULA ( T, T3 )
        CALL MATMULA ( T, T4 )
        CALL MATMULA ( T, T5 )
        CALL MATMULA ( T, T6 )

C Calculate the "nominal" wire length based on current parameter values

        CALL LENGTH(OTTPOP,T)

C Calculate the function F

        F(J)=DABS(OTTPOP-OTX(J))

C End the do-loop for counter J

      ENDDO

C Compute RMS error

      SUMSQ=0.0D0
      DO J=1,NOBS
        SUMSQ=SUMSQ+F(J)*F(J)
      ENDDO
```

```
      RMS=DSQRT(SUMSQ/NOBS)
      WRITE(6,*)RMS
      RETURN
      END

C ***********************************************************************

C This subroutine calculates the length of wire from the base fixture to
C the manipulator endpoint fixture based on the current manipulator
C endpoint pose (T6 = T). This subroutine uses a renamed version of IMSL
C routine ZXSSQ (ZXSSQ1) to minimize the sum (component by component) of
C unit vectors describing the tangent points for both upper and lower
C fixtures. Subroutine ZXSSQ1 utilizes subroutine MINLENTH to evaluate
C the "F" functions.

      SUBROUTINE LENGTH(OTTPOP,T)
      REAL*8 T6(4,4),T6INV(4,4),T(4,4)
      REAL*8 EPSN,DELTAN,PARMN(4),XN(4),SSQN,FN(6),XJACN(6,4)
      REAL*8 WORKN(42),XJTJN(10),XUUVP(4)
      INTEGER MNN,NNN,NSIGN,MAXFNN,IOPTN,IXJACN,INFERN,IERN
      REAL*8 PI,R,X,Y,Z,OOP,XLUV(4),XUUV(4),XO,YO,P6(4),XOYO
      REAL*8 X6Y6,ZO,OTTPOP,OTSCAL,OT,XU(4),TTP,TTPSCAL
      REAL*8 THETAU,THETAL
      COMMON/LEN/ PI,R,T6,THETAU,THETAL,TTP

      EXTERNAL MINLENTH
      DO I=1,4
        DO J=1,4
          T6(I,J)=T(I,J)
        ENDDO
      ENDDO
      MNN=6
      NNN=4
      NSIGN=4
      EPSN=0.0D0
      DELTAN=0.0D0
      MAXFNN=1000
      IXJACN=6
      IOPTN=1
      PI=4.0D0*DATAN(1.0D0)
      R=12.70D0

      X=T6(1,4)
      Y=T6(2,4)
      Z=T6(3,4)
      OOP=DSQRT(X**2+Y**2+Z**2)
      XLUV(1)=X/OOP
      XLUV(2)=Y/OOP
      XLUV(3)=Z/OOP
      XLUV(4)=1.0D0
      DO I=1,4
```

```
        DO J=1,4
          T6INV(I,J)=0.0D0
        ENDDO
      ENDDO
      DO I=1,3
        DO J=1,3
          T6INV(I,J)=T6(J,I)
        ENDDO
        T6INV(1,4)=T6INV(1,4)-T6(I,4)*T6(I,1)
        T6INV(2,4)=T6INV(2,4)-T6(I,4)*T6(I,2)
        T6INV(3,4)=T6INV(3,4)-T6(I,4)*T6(I,3)
      ENDDO
      T6INV(4,4)=1.0D0
      XUUVP(1)=X-XLUV(1)
      XUUVP(2)=Y-XLUV(2)
      XUUVP(3)=Z-XLUV(3)
      XUUVP(4)=1.0D0
      DO I=1,4
        XUUV(I)=0.0D0
        DO J=1,4
          XUUV(I)=XUUV(I)+T6INV(I,J)*(XUUVP(J))
        ENDDO
      ENDDO
      XN(1)=XLUV(1)*R
      XN(2)=XLUV(2)*R
      XN(3)=XUUV(1)*R
      XN(4)=XUUV(2)*R



C Call renamed version of ZXSSQ

      CALL ZXSSQ1(MINLENTH,MNN,NNN,NSIGN,EPSN,DELTAN,MAXFNN,IOPTN,
     &     PARMN,XN,SSQN,FN,XJACN,IXJACN,XJTJN,WORKN,INFERN,IERN)

C Calculate wire length

      OTTPOP=(THETAL+THETAU)*R+TTP

      RETURN
      END


C *********************************************************************

      SUBROUTINE MINLENTH(XN,MNN,NNN,FN)

      INTEGER MNN,NNN
      REAL*8 XN(4),FN(6),ERROR,AAA,BBB
      REAL*8 PI,R,X,Y,Z,OOP,XLUV(4),XUUV(4),XO,YO,P6(4),XOYO
      REAL*8 X6Y6,ZO,OPTPSCAL,OPT,OTSCAL,OT,XU(4),TTP,TTPSCAL
```

```
      REAL*8 T6(4,4),P6P(4),THETAU,THETAL,MAGUL,MAGUU,UX0
      REAL*8 UY0,UZ0,XUP(4),UX6,UY6,UZ6,UXT,UYT,UZT,Z0P

      COMMON/LEN/ PI,R,T6,THETAU,THETAL,TTP

      X0=XN(1)
      Y0=XN(2)
      P6(1)=XN(3)
      P6(2)=XN(4)
      P6P(1)=P6(1)
      P6P(2)=P6(2)
      X0Y0=DSQRT(X0**2+Y0**2)
      X6Y6=DSQRT(P6(1)**2+P6(2)**2)
      AAA=2.0D0*R*X6Y6-P6(1)**2-P6(2)**2
      BBB=2.0D0*R*X0Y0-X0**2-Y0**2
      IF (AAA .LT. 0.0D0) THEN
        P6(3)=R
      ELSE
        P6(3)=DSQRT(AAA)
      ENDIF
      IF (BBB .LT. 0.0D0) THEN
        Z0=R
      ELSE
        Z0=DSQRT(BBB)
      ENDIF
      THETAU=DASIN(P6(3)/R)
      THETAL=DASIN(Z0/R)
      P6P(3)=X6Y6*DTAN(PI/2.0D0-THETAU)
      Z0P=X0Y0*DTAN(PI/2.0D0-THETAL)
      MAGUL=DSQRT(X0**2+Y0**2+Z0P**2)
      MAGUU=DSQRT(P6P(1)**2+P6P(2)**2+P6P(3)**2)
      UX0=X0/MAGUL
      UY0=Y0/MAGUL
      UZ0=Z0P/MAGUL

C Transform tangent vector in FE coordinates to FM coordinates

      P6P(4)=1.0D0
      DO I=1,4
        XUP(I)=0.0D0
        DO J=1,4
          XUP(I)=XUP(I)+T6(I,J)*P6P(J)
        ENDDO
      ENDDO

C Calculate upper tangent vector in FM coordinates

      UX6=(XUP(1)-T6(1,4))/MAGUU
      UY6=(XUP(2)-T6(2,4))/MAGUU
      UZ6=(XUP(3)-T6(3,4))/MAGUU
      P6(4)=1.0D0
```

```
      DO I=1,4
        XU(I)=0.0D0
        DO J=1,4
          XU(I)=XU(I)+T6(I,J)*P6(J)
        ENDDO
      ENDDO
      TTP=DSQRT((XU(1)-X0)**2+(XU(2)-Y0)**2+(XU(3)-Z0)**2)
      UXT=(XU(1)-X0)/TTP
      UYT=(XU(2)-Y0)/TTP
      UZT=(XU(3)-Z0)/TTP

C Calculate minimizing functions

      FN(1)=UXT-UX0
      FN(2)=UYT-UY0
      FN(3)=UZT-UZ0
      FN(4)=UXT+UX6
      FN(5)=UYT+UY6
      FN(6)=UZT+UZ6

C Calculate residual

      ERROR =0.0D0
      DO I=1,6
        ERROR=ERROR + FN(I)**2
      ENDDO
      WRITE(6,*)ERROR

      RETURN
      END
```

APPENDIX F

```
C ***********************************************************************

      PROGRAM WVERIFY

C This program generates the six-dof pose  error for the PUMA manipulator.
C It contains the identified calibration parameters and the exact parameter.
C It uses a data file of verification joint angle sets POSEVER.DAT, and the
C file RESULT.DAT from the program ID6.

      INTEGER I, J, K, NPOSES, N
      REAL*8 DANGLE, DLENTH
      REAL*8 DT(5),DD(5),AA(5),AL(5),BL(5),MEAS(6)
      REAL*8 EDT(5),EDD(5),EAA(5),EAL(5),EBL(5),EMEAS(6)
      REAL*8 EDF6,EFI6,ETH6,ESI6,EPX6,EPY6,EPZ6
      REAL*8 THETA(1000,6), TDELTA(4,4)
      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4)
      REAL*8 T4(4,4), T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4), ET(4,4)

      REAL*8 DT1, DT2, DT3, DT4, DT5
      REAL*8 DD1, DD2, DD3, DD4, DD5
      REAL*8 AA1, AA2, AA3, AA4, AA5
      REAL*8 AL1, AL2, AL3, AL4, AL5
      REAL*8 BL1, BL2, BL3, BL4, BL5
      REAL*8 DF6, FI6, TH6, SI6, PX6, PY6, PZ6
      REAL*8 XW, YW, ZW
      COMMON TIMAT,THETA

C Initialize the TIMAT matrix to an I matrix:

      DATA TIMAT/1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1/

C Open data file

      OPEN (9,  NAME='POSEVER.DAT',STATUS='OLD')
      OPEN (10, NAME='INPUT.DAT', STATUS='OLD')
      OPEN (11, NAME='RESULT.DAT', STATUS='OLD')

C Read input parameters

      READ (10,*)
      READ (10,*) MEAS(1),MEAS(2),MEAS(3),MEAS(4),MEAS(5),MEAS(6)
      READ (10,*) DT1,DD1,AA1,AL1,BL1
      READ (10,*) DT2,DD2,AA2,AL2,BL2
      READ (10,*) DT3,DD3,AA3,AL3,BL3
```

```
      READ (10,*) DT4,DD4,AA4,AL4,BL4
      READ (10,*) DT5,DD5,AA5,AL5,BL5
      READ (10,*
      READ (10,*) DF6,TH6,SI6,PX6,PY6,PZ6
      READ (10,*)
      READ (10,*) NOBS,R,DANGLE,DLENTH,MAGNX,MAGNL

      CLOSE (10)


C Read in joint angle sets for verification poses

      NPOSES=NOBS

      DO I=1,NPOSES
        READ(9,*)
        READ(9,*)THETA(I,1),THETA(I,2),THETA(I,3),THETA(I,4),
     &           THETA(I,5),THETA(I,6)
      ENDDO
      CLOSE(9)

C Set exact link parameters for the manipulator:

      DO I=2,5
        DT(I)=DT(I)+DANGLE
      ENDDO

      MEAS(1)=MEAS(1)
      MEAS(2)=MEAS(2)
      MEAS(3)=MEAS(3)
      MEAS(4)=MEAS(4)+DLENTH
      MEAS(5)=MEAS(5)+DLENTH
      MEAS(6)=MEAS(6)+DLENTH

      AL(1)=AL1+DANGLE
      AL(2)=AL2+DANGLE
      AL(3)=AL3+DANGLE
      AL(4)=AL4+DANGLE
      AL(5)=AL5+DANGLE

      AA(1) = AA1 + DLENTH
      AA(2) = AA2 + DLENTH
      AA(3) = AA3 + DLENTH
      AA(4) = AA4 + DLENTH
      AA(5) = AA5 + DLENTH

      DD(1) = DD1
      DD(2) = DD2
      DD(3) = DD3 + DLENTH
      DD(4) = DD4 + DLENTH
      DD(5) = DD5 + DLENTH
```

```
      BL(1) = BL1
      BL(2) = BL2 + DANGLE
      BL(3) = BL3
      BL(4) = BL4
      BL(5) = BL4

      DF6 = DF6 + DANGLE
      TH6 = TH6
      SI6 = SI6
      PX6 = PX6 + DLENTH
      PY6 = PY6
      PZ6 = PZ6 + DLENTH

C Read in and set up estimated parameter table

      READ(11,*)
      READ(11,*)
      READ(11,*) EMEAS(1),EMEAS(2),EMEAS(3)

      DO I=1,5
        READ(11,*)
        READ(11,*)
        READ (11,*) EDT(I),EDD(I),EAA(I),EAL(I),EBL(I)
      ENDDO

      READ(11,*)
      READ(11,*)
      READ(11,*) EDF6,ETH6,ESI6,EPX6,EPY6,EPZ6

C Main loop through NPOSES joint angle sets

      DO K=1,NPOSES

        CALL FKS (K,MEAS,DT,AL,AA,DD,BL,FI6,TH6,SI6,PX6,PY6,PZ6,T)
        CALL FKS (K,EMEAS,EDT,EAL,EAA,EDD,EBL,EFI6,ETH6,ESI6,EPX6,
     &            EPY6,EPZ6,ET)

C Compute the differential tool matrix

        CALL MATSUB(TDELTA,T,ET)

c Compute the pose errors

        POSERR=SQRT(TDELTA(1,4)**2+TDELTA(2,4)**2+TDELTA(3,4)**2)
        ORERR1=(TDELTA(3,2)-TDELTA(2,3))/2
        ORERR2=(TDELTA(1,3)-TDELTA(3,1))/2
        ORERR3=(TDELTA(2,1)-TDELTA(1,2))/2
        ORERR=SQRT(ORERR1**2+ORERR2**2+ORERR3**2)

c Update total error counts
```

```
        POSTERR=(POSERR+(K-1)*POSTERR)/K
        ORTERR=(ORERR+(K-1)*ORTERR)/K

c End of main loop

      ENDDO

      WRITE(6,*) 'Position error, orientation error'
      WRITE(6,*) POSTERR,ORTERR
      END

c *****************************************************************

      SUBROUTINE FKS (N,MEAS,DT,AL,AA,DD,BL,DF6,TH6,SI6,
     &                PX6,PY6,PZ6,T)

      REAL*8 T0(4,4), T1(4,4), T2(4,4), T3(4,4)
      REAL*8 T4(4,4), T5(4,4), T6(4,4), TRPY(4,4), TXYZ(4,4)
      REAL*8 TIMAT(4,4), T(4,4), dt(5),al(5),aa(5),dd(5),bl(5)
      REAL*8 THETA(1000,6),ANG(5),MEAS(6)

      COMMON TIMAT,THETA

C Initialize the T matrix to an I matrix:

      DO J=1,4
        DO K=1,4
          T(J,K) = TIMAT(J,K)
        ENDDO
      ENDDO

C Set up the joint angles

      DO I=1,5
        ANG(I)=THETA(N,I)+DT(I)
      ENDDO

      FI6=THETA(N,6)+DF6

C Compute the T matrices, T1 thru T6:

      CALL T3RPY (MEAS(1),MEAS(2),MEAS(3),T0)
      CALL T3XYZ (MEAS(4),MEAS(5),MEAS(6),T0)
      CALL MATMULC (T0,TRPY,TXYZ)

      CALL TRANSFORM (AL(1),AA(1),DD(1),ANG(1),BL(1),T1)
      CALL TRANSFORM (AL(2),AA(2),DD(2),ANG(2),BL(2),T1)
      CALL TRANSFORM (AL(3),AA(3),DD(3),ANG(3),BL(3),T1)
      CALL TRANSFORM (AL(4),AA(4),DD(4),ANG(4),BL(4),T1)
      CALL TRANSFORM (AL(5),AA(5),DD(5),ANG(5),BL(5),T1)
```

```
      CALL T3RPY (FI6,TH6,SI6,TRPY )
      CALL T3XYZ (PX6,PY6,PZ6,TXYZ )
      CALL MATMULC (T6,TRPY,TXYZ )

C Compute the overall transformation, T:

      CALL MATMULA ( T, T0 )
      CALL MATMULA ( T, T1 )
      CALL MATMULA ( T, T2 )
      CALL MATMULA ( T, T3 )
      CALL MATMULA ( T, T4 )
      CALL MATMULA ( T, T5 )
      CALL MATMULA ( T, T6 )

      RETURN
      END

c ***************************************************************************
```

APPENDIX G

Two of the more important aspects in robot calibration are precise measurements and maximum joint excursions during data collection. An instrument referred to as a linear slide is capable of displacement measurements accurate to 0.01 mm. However, restricting the end effector to linear travel can severely limit joint variation for one or more manipulator joints. In fact, there are a number of configurations in which one joint may not vary at all. The purpose of this project then is to establish a position and orientation of the slide which will maximize joint excursion for all six joints. The project will require a dual use of the ADS program as is described below.

A number of methods exist for developing an analytical approach to the forward kinematic solution for a manipulator (ie given a set of joint angles, what is the position and orientation (pose) of the end effector). However, an analytical solution to the inverse kinematics is much more difficult or impossible to develop. Therefore, the first application of ADS will be to solve this nonlinear problem in the following manner. The design variables will be the 6 joint angles. The design variables are bounded by the physical bounds on their rotations and additional restrictions to limit the robot to one "arm configuration". The design variables are

used to produce a forward kinematic solution which is then compared to the "desired" pose. The goal then is to minimize the error between the "desired" and calculated poses. The pose information is in the form of a four by four homogeneous transformation matrix which is calculated in the following manner. A coordinate frame is assigned to each manipulator link in a standardized method. Due to two geometric constraints, 4 parameters are required to transform from link to link. These parameters include two rotations of which one is the rotary joint angle and the other is a twist angle, and two translations which are essentially the link length and an offset distance. Using a standardized approach, the transformation takes the following form:

$$T_{i-1}^{i} = \begin{bmatrix} a_1 & b_1 & c_1 & x \\ a_2 & b_2 & c_2 & y \\ a_3 & b_3 & c_3 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} = f(a_i, d_i, \theta_i, \alpha_i)$$

where the a_i, b_i and c_i entries are direction cosines and x, y, and z entries are the position with respect to the i-1 coordinate frame.

As noted, the transformation is a function of the four parameters and in this application the nominal values are used for the twist angle and the translations. The rotary joint angle is the variable for each transformation matrix. If the

frame to frame transformation matrices are multiplied in the following manner, then the manipulator pose (ie. the position and orientation of the manipulator end effector with respect to a "world" coordinate frame will be given by the T^6 matrix:

$$T^6 = T_W^0\ T_O^1\ T_1^2\ T_2^3\ T_3^4\ T_4^5\ T_5^6$$

To compute the inverse solution, a desired pose matrix is formed which is facilitated in this application by orientating the "world" coordinate frame with the axis of the slide and at its zero position. The design variables (the six joint angles) are given an initial value which are used to compute a forward solution. The difference between the computed T^6 matrix and desired T^6 matrix are calculated term by term. Then, the objective function is formed as the sum of the squares of the element by element differences. Note that if a solution exists (ie. reachable by the manipulator), the objective function will be zero (or at least "small"). This objective function value will then be used as a constraint in the second application of the ADS program to ensure the slide is "reachable" by the robot. The following is a mathematical statement of the problem.

First application of ADS:

Design variables:

$$\bar{X} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix}$$

Minimize:

$$F(\bar{X}) = \sum \delta_{i,j} \qquad i,j=1,2,3,4$$

where

$$\delta_{i,j} = |d_{i,j} - c_{i,j}|$$

and $d_{i,j}$ and $c_{i,j}$ are the i,j^{th} entries of the "desired" and calculated transformation matrices.

In the second application of the ADS program, the design variables are the x,y and z position of the end of the slide as well as two orientation angles theta and phi which are azimuth and elevation angles. The end of the slide must lie outside a circle of radius of 150 mm from the joint one axis of rotation and this becomes one of the nonlinear inequality constraints. The inverse solution for six poses along the

slide will be computed as described above which will serve two purposes. As stated earlier, it will constrain the problem within the "reachable" range of the manipulator. This will make up six nonlinear inequality constraints. Additionally, the inverse solution provides six sets of joint angles for one slide position and orientation. The maximum joint excursion for each joint is then determined from this information. The goal is to maximize the excursion of all six joint angles. Therefore, objective function will be to minimize the negative value of joint one range over slide travel. The additional five joint excursion ranges are compared to joint one's range and these form 5 additional linear inequality constraints. Additionally, the maximum displacement of the slide zero point was placed at 1000 mm from the base frame of the robot and this made up the thirteenth constraint.

M a t h e m a t i c a l l y s t a t e d :

Second application of ADS:

Design variables:

$$\overline{X} = \begin{bmatrix} x \\ y \\ z \\ \theta \\ \phi \end{bmatrix}$$

Minimize:

$$F(\bar{x}) = -\Delta\theta_1$$

where delta theta is the maximum range of joint excursion for the six positions along the slide

Subject to:

$$\begin{aligned}
g_1 &= \Delta\theta_1 - \Delta\theta_2 \leq 0 \\
g_2 &= \Delta\theta_1 - \Delta\theta_3 \leq 0 \\
g_3 &= \Delta\theta_1 - \Delta\theta_4 \leq 0 \\
g_4 &= \Delta\theta_1 - \Delta\theta_5 \leq 0 \\
g_5 &= \Delta\theta_1 - \Delta\theta_6 \leq 0 \\
g_6 &= F_1 - 0.01 \leq 0 \\
g_7 &= F_2 - 0.01 \leq 0 \\
g_8 &= F_3 - 0.01 \leq 0 \\
g_9 &= F_4 - 0.01 \leq 0 \\
g_{10} &= F_5 - 0.01 \leq 0 \\
g_{11} &= F_6 - 0.01 \leq 0 \\
g_{12} &= \frac{\sqrt{x^2+y^2} - 150.0}{150.0} \leq 0 \\
g_{13} &= \frac{\sqrt{x^2+y^2+z^2} - 1500}{1500} \leq 0
\end{aligned}$$

where F_i is the value of the objective function for each of the six positions along the slide as calculated in the first application of ADS (inverse kinematic solution) which will be referred to as the "inner loop" of the program.

The first application of ADS was tested independently. After considerable testing, it was found that the following combination of methodology and parameter settings performed "best".

istrat	0
iopt	3
ioned	3
dabobj	0.001
dabobm	0.0001
dabstr	0.0001
fdch	0.0001
fdchm	0.00001
itmax	60
scaling	off

The results of these tests were compared to results obtained from a well tested IMSL routine which was unsuitable for implementation within the "inner" loop of the main program due to the inability to bound the joint angles and obtain an objective function value when the endpoint was not "reachable". The chosen method consistently converged with an accuracy within 3 digits of the IMSL routine. The major drawback was the average of 200 function evaluations required. However, accuracy and precision were crucial so the large number of function calls was a necessary tradeoff.

With the first ADS application developed and working, this program was then implemented as a subroutine within the second application of ADS. This program was tested using the recommended methods of the ADS manual and additionally with

one dimensional search methods 2 and 6. No matter what strategy or optimizer chosen, ioned values of 2 and 6 always failed to converge (there is no obvious reason for this other than the non-linearity of the problem). Additionally, strategies 3, 6 and 7 failed in all configurations attempted (ie parameter setting, optimizer, and one dimensional search variations). Of the variations tested that converged, the following combination of methodology and parameter settings worked "best".

istrat = 9
iopt = 5
ioned = 7
scaling off
dabobj = 0.001
dabobm = 0.0001
dabstr = 0.0001

"Best" in this setting was determined by convergence and number of function calls. This best method consistently converged in the fewest number of function calls. However, even though this is the fastest method, the program still required as much as 15-20 minutes of run-time on a VAX 3100 station (30 minutes or longer on the older workstations). As expected (due to the nonlinearity of the problem), there are apparently a number of local minima. Several starting positions were chosen and tested. Printouts of the results for several initial values are enclosed. The following lists the optimum of the local minima found. From the geometry of the problem this result is probably "close" to the global minimum

and was a significant improvement from the initial design values and the design used in the actual experiment.

One improvement to the program which significantly improved convergence and reduced run-time was a seeding method used for the inner loop. A reasonable set of initial joint angles was chosen prior to the first pass through the subroutine. The joint angles calculated for the zero position on the slide were then used for initial values for calculating the joint angles of the second position. This process was repeated for each additional measurement position. After the main program design variables are varied, the initial value of the joint angles are set to the previously calculated joint angles for the previous slide zero position before calling the subroutine.

Additional testing of both parameter settings and initial values should be performed to both ensure that the program performs "optimally" and to search for the global minimum. However, the results at this stage are quite satisfactory.

RESULTS

$\Delta\Theta_1 = 42.8$
$\Delta\Theta_2 = 46.6$
$\Delta\Theta_3 = 96.8$
$\Delta\Theta_4 = 80.1$
$\Delta\Theta_5 = 42.8$
$\Delta\Theta_6 = 45.7$

$x = 77.9$ *mm*
$y = -431$ *mm*
$z = 248$ *mm*
$\theta = 193°$
$\phi = 20.4°$

$x_0 = -200$ *mm*
$y_0 = -200$ *mm*
$z_0 = 400$ *mm*
$\theta_0 = 135°$
$\phi_0 = 45°$

FUNCTION EVALUATIONS FOR MAIN LOOP: 120

FUNCTION EVALUATIONS PER INNER LOOP ITERATION: AVERAGE OF 200.

LIST OF REFERENCES

1. Driels, M.R., Swayze, W.E., and Potter, S.A., "Full Pose Claibration of a Robot Manipulator Using a Coordinate Measuring Machine," Submitted for Publication, p. 1, 1991.

2. Driels, M.R., Swayze, W.E., and Potter, S.A., "Full Pose Claibration of a Robot Manipulator Using a Coordinate Measuring Machine," Submitted for Publication, p. 1, 1991.

3. Driels, M.R., "Using Passive End-Point Motion Constraints to Calibrate Kinematic Mechanisms," Submitted for Publication, 1991.

4. Paul, R.P., *Robot Manipulators: Mathematics, Programming and Control*, p. 25, The MIT Press, 1982.

5. Paul, R.P., *Robot Manipulators: Mathematics, Programming and Control*, p. 53, The MIT Press, 1982.

6. Mooring, B.W., Roth, Z.S., and Driels, M.R., *Fundamentals of Manipulator Calibration*, p. 45, John Wiley and Sons, Inc., 1991.

7. Hayati, S. and Mirmirani, M., "A Software for Robot Geometry Parameter Estimation", SME Paper #MS84-1052, presented at Robots West Conference, Anaheim, California, November, 1984.

8. Mooring, B.W., Roth, Z.S., and Driels, M.R., *Fundamentals of Manipulator Calibration*, p. 43, John Wiley and Sons, Inc., 1991.

9. Paul, R.P., *Robot Manipulators: Mathematics, Programming and Control*, p. 90, The MIT Press, 1982.

10. Paul, R.P., *Robot Manipulators: Mathematics, Programming and Control*, pp. 22-23, The MIT Press, 1982.

11. Potter, S.A., *Full Pose and Partial Pose Calibration of a Six Degree of Freedom Robot, Manipulator Arm*, Master's Thesis, Naval postgraduate School, Monterey, California, September, 1991, pp. 55-65.

12. Driels, M.R., *Using Passive End-Point Motion Constraints to Calibrate Kinematic Mechanisms*, Submitted for Publication to ASME, 1990.

13. Potter, S.A., *Full Pose and Partial Pose Calibration of a Six Degree of Freedom Robot, Manipulator Arm*, Master's Thesis, Naval Postgraduate School, Monterey, California, September, 1991, p. 75.

14. Pathre, U.S. and Driels, M.R., "Simulation Experiments in Parameter Identification for Robot Calibration", *The International Journal of Advanced Manufacturing Technology*, v. 4, pp.25-32, 1990.

15. Mooring, B.W., Roth, Z.S., and Driels, M.R., *Fundamentals of Manipulator Calibration*, p. 42-43, John Wiley and Sons, Inc., 1991.

16. Whitney, D.E., Lozinski, C.A., and Poncke, J.M., "Industrial Robot Forward Calibration Method and Results", *Journal of Dynamic Systems, Measurement and Control*, 108(1): 1-8, March, 1986.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Naval Engineering Curricular Office Code 34 Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	1
3.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5000	2
4.	Department Chairman, Code ME Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	1
5.	Professor Morris Driels, Code ME/Dr Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	3
6.	LT William E. Swayze 3547 NE Quinault Drive Bremerton, Washington 93940	3

Thesis
S92693 Swayze
c.1 Modelling experimental procedures for manipulator calibration.